中華學習機
CEC－2000

用戶使用手册

中 國 教 育 電 子 公 司

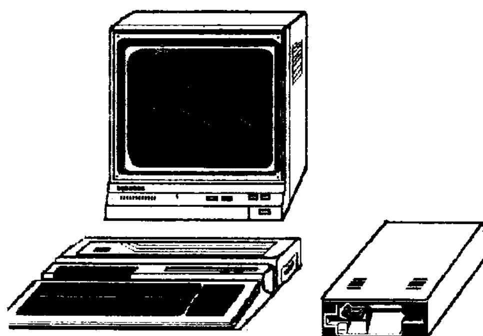
平面設計:李杰

中 國 教 育 電 子 公 司

中華學習機CEC-2000用戶使用手册

中國教育電子公司

CEC—2000 中华学习机
使用手册



中国教育电子公司
1991

第一篇　CEC—2000 中华学习机简介及安装方法

第一章

1.1 系统简介

CEC—2000 中华学习机是 CEC—I 中华学习机的增强型。它除了具有 CEC—I 的系统功能外，增加了许多新的功能，比如：ROM、RAM 均可以扩充，提高了图形显示分辨率等，此外还扩展了一块游戏机控制卡，使得 CEC—2000 中华学习机兼有微型计算机及游戏机的双重身份。使用户能一举两得。

一套 CEC—2000 中华学习机主要由三部分组成：主机、显示器和外存储器。其中主机是以 86 键的键盘为外壳的。显示器可以用彩色电视机，也可以用监视器。外存储器包括：盒式录间机、软盘驱动器。可以根据用户的不同要求，使用不同的配件安装自己的系统。

1.1.1　键盘

CEC—2000 中华学习机的键盘与 CEC—I 一样仍安装在主机外壳上，与主机构成一体化结构。所不同的是增加了小键盘区，键体总个数达到 86 键，除 CEC—2000 上的"Bause"键等效于 CEC—I 的"QUIT"键外，其它键的定义均相同。

1.1.2　显示器

CEC—2000 中华学习机采用以复合视频信号方式工作的单色或 NTSC/PAL 两种制式的彩色监视器。此外，通过电视接口可以连接黑白或 NTSC/PAL 两种制式的彩色电视机作

为系统的显示器。

1.1.3　盒式录音机

CEC—2000 中华学习机保留了原 CEC—I 中华学习机上的录音机接口，可以用盒式录音机作为外存储器。

1.1.4 软盘驱动器

软盘驱动器是最常用的外存储器，用来存放程序、数据等。CEC—2000 中华学习机可以连接两台 5.25 英寸单面单密度软盘驱动器，每台容量 143K 字节。

1.1.5 主板机

CEC—2000 的电路部分做在盒内主机板上。包括中央处理器 6502，存储器电路，以及输入输出接口电路。系统的电源安装在主机盒内。

中央处理器 6502 为八位微处理器。时钟频率为 1MHZ，寻址范围 64K 字节。

存储器由 RAM 和 ROM 两部分组成。RAM 分为主存储器和辅存储器，主存储器容量为 64K 字节，辅存储器可通过插内存扩展卡进步扩充。ROM 容量为 64 字节，由 2 片 EPROM27256 组成。用户可根据需要换成 2 片 EPROM27512，使 ROM 的容量扩大到 128K 字节。

主板上使用了几片专用大规模集成电路(ASIC)，其中包括存储器管理部件 MMU，键盘编码电路 KBC，软盘控制器 FDC，彩色编码电路 CEU，输入输出部件 IOU，以及可编程阵列逻辑电路。

主机的左侧有一个与 Apple Ⅱ 兼容的 50 线输入输出插座，用于扩充功能。输入输出接口电路和电源在下面说明。此外 CEC—2000 还可以插接游戏机控制卡和内存扩展卡。

1.1.6　输入输出接口

CEC—2000 中华学习机的输入输出接口电路包括键盘输入接口，显示器输出接口，盒式录音机输入输出接口，软盘驱动器接口，打印机接口，以及游戏控制杆接口。

1.1.7 电源

主机盒内装有一个开关电源，输出＋5V，＋12V，－5V，－12V 四种直流电源，最大交流功耗为 45W，各路电压的额定负载电流为：

＋5V	2.3A
＋12V	1.3A
－5V	0.1A
－12V	0.1A

1.2 主要技术指标

一、主机

1. 中央处理器：6502，8 位微处理器

2. 内存容量：

RAM：64K 字节。可以再扩充 256K 字节。

ROM：64K 字节，其中固化监控程序，CEC—BASIC 语言及汉字管理程序、自检程序。可扩充至 128K 字节。

3. 采用几种专用大规模集成电路：分别为 MMU 存储器管理部件，IOU 输入/输出管理部件，CEU 彩色编码电路，FDC 软盘控制器，KBC 键盘编码电路，以及时序电路。

4. 汉字系统：汉字系统已做在主机板上。该系统提供拼音，区位，音形等输入方式，采用全点阵的国标一、二级汉字字库(两片一兆位 ROM)，包括 6763 个汉字及外文字母等。

5. 显示器接口

一个是复合视频信号输出，可连接 NTSC/PAL 制式彩色或单色监视器。另一个是射频信号输出，复合视频信号经过带

声音输入的射频调制器可连接 NTSC/PAL 两种制式的彩色或黑白电视机。

6. 软盘驱动器接口

CEC—2000 中华学习机可以直接连接两台 5.25 英寸单面单密度软盘驱动器。

7. 盒式录音机接口:可与一般家用或专用录音机连接,接口电路输出 25mv 电压信号作为录入信号,要求录音机的读出信号≥1V(峰—峰值),电路的输出阻抗为 100Ω,输入阻抗为 125Ω。

8. 键盘接口:键盘为标准打字机键盘,共 86 键,包括字母、数字及一些特殊控制功能键,采用弹簧键体。

9. 并行打印机接口:可连接具有 Centronic 标准接口的打印机。

10. 游戏控制杆接口:是一个两用接口,可连接 CEC—2000 中华学习机专用游戏控制杆或任天堂游戏机用冲峰枪等。并提供三个 TTL 电平的开关量输入,四个模拟量输入,可连接 CEC—I 中华学习机用游戏控制杆,另外在主板上有一个 16 线 IC 插座,提供游戏杆控制信号以及四个开关量输出信号。

11. 扩充槽口:提供 1 个与 Apple Ⅱ 兼容的 50 线 I/O 插座。

12. 游戏机控制卡接口:在机内该接口上插上 CEC—2000 中华学习机专用 GAME 卡后,可使中华学习机作为游戏机使用。

13. 内存扩展卡:这也是一个机内接口。当专用内存扩展卡插在主机板上后,内存容量最大能扩展到 320K 字节。

14. 主机盒内装有开关电源,最大交流功耗为 45W。

二、显示器

可以使用具有复合视频信号输入的 NTSC/PAL 制式彩色、单色监视器或电视机。

屏幕显示功能：

1. 西文方式

字符构成：5×7 点阵

每帧字符：40 字符×24 行

80 字符×24 行(需配有内存扩展卡)

显示方式：正常、反相、闪烁(仅适用于每行 40 个字符)

2. 中文方式

字符构成：16×16 点阵(汉字)

8×16 点阵(ASCⅡ)

每帧字符：每帧为 11 行，外加 1 行提示行。

每行字符数：17(汉字)及 34(ASCⅡ)个

3. 图形方式

低分辩率彩色图形显示，分辩率为 40H×48V，16 种颜色。

高分辩率彩色图形显示，分辩率为 280H×192V，6 种颜色。

倍高分辩率彩色图形显示，分辩率为 560H×192V，16 种颜色。(要配有内存扩展卡)

三、软盘驱动器

可采用 1～2 台 5.25 英寸单面单密度软盘驱动器。

四、盒式磁带机

任选一般盒式录音机或专用录音机。

五、打印机

可直接与具有 Centronic 接口标准的打印机连接，如 CP

—80,MX—80,RX—80,FX—80 等九针打印机。

六、软件

系统软件:监控程序,DOS3.3,CEC—BASIC 以及汉字管理软件。

应用软件:多种教学辅助软件和游戏程序以及和 CEC—I、Apple Ⅱ 兼容的应用软件。

第二章　操作指南

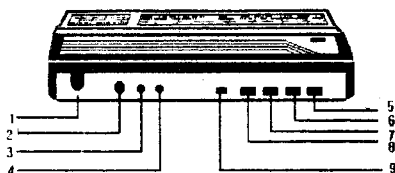
2.1 系统安装

2.1.1 主机

主机是一个结构紧凑的长方形机盒。机盒上面前半部为键盘,右上角并列两只红色指示灯。右侧为电源指示灯,左侧为大写字锁定指示灯。开机后,电源指示灯即亮,大写字锁定指示灯也处于亮的位置。此灯亮表示输入的英文字符为大写字符,不亮表示输入的英文字符为小写字符,通过按下 Caplock 键可以改变其状态。机盒左侧为扩展槽口,右侧为电源开关,后侧为各种接口。

一、主机接口

主机的接口参照图 2—1



1	POWER(~220)	电源
2	TYPE RECORD	录音机插座
3	MONITOR	监视器插座
4	TVVIDEO	音频、视频外接口
5	DRIVE1	驱动器 A 接口
6	DRIVE2	驱动器 B 接口
7	PRINTER	打印机接口插座
8	COMPRT&GAME	计算机上游戏机转接开关
9	JOYSTICK2	游戏机游戏杆接口

图 2—1

位于机盒左侧，有一 50 芯扩展槽口，如图 2—2 所示。

二、内部开关

图 2—3 为机盒内主机板上开关位置图

J17—黑白/彩色转换，短接为彩色方式。

J18—使用监视器或电视机为 NTSC 制式时，将此插针短接，选用 PL 制式，将此断开。

J16—机内喇叭接口，此两针之间可接一只 8Ω 扬声器

J19—扩充槽口号的选择，用二个插针完成这项选择，我们把它们编为 1 号插针，2 号插针。

1 号插针插 1，2 孔，2 号插针插 3，4 孔，选择 2 号槽

1 号插针插 5，6 孔，2 号铺针插 7，8 孔，选择 4 号槽

J15—机内游戏戏控制插座

2.12 显示器的联接

CEC—2000 中华学习机可以选用以下任一种显示器：NTSC 或 PAL 制彩色电视机、监视器或单色电视机、监视器。

一、电视的联接

步聚如下：

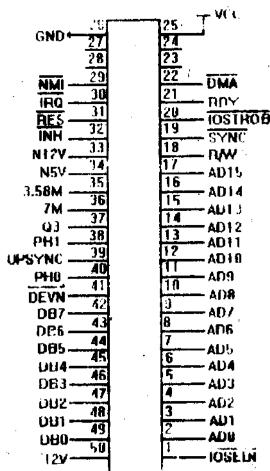


图 2—2

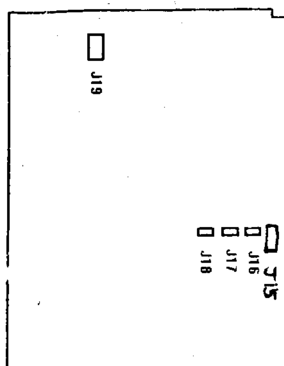


图 2—3

1. 将主机和电视放在桌上，并备两个电源插座。

2. 把电视机的天线插头去掉。

3. 用视频电缆线一头插入主机电视机接口，一头插入电视机天线插座，电视机天线插座的输入阻抗为 75Ω。

4. 确认主机电源开关处于“OFF”位置。

5. 将电视机电源插头和主机电源插头分别插入电源插座。见图 2—4。

6. 把电视机音量开关拧到最小。

7. 先开电视再开主机。

8. 当使用黑白电视机时，首先调准电视机频道(以出厂厂家要求为准。)开机后，调节频道微调及对比度、亮度旋纽，直到屏幕出现清晰的“ZHONG HUA XUE XI JI”字样。

当使用彩色电视机时，同样先选好一个频道，并通过自检

190 POKE −16298,0

改变制图页，从一个高分辨率制图的页转变到正文的同样页上，不清除屏幕。当你从 BASIC 进入到整型 BASIC 时，这是必须的。否则，整型 BASIC 的 GR 指令可能错误地显示高分辨率页。

依赖于其它开关的置位，这可以使得显示从高分辨率制图的第一页改变为低分辨率制图的第一页；从高分辨率制图的第二页改变到低分辨率制图的第二页；或者（在正文方式下）可以使得显示不改变。

195 POKE −16297,0

对图形换页，从一个文本显示页到高分辨率的相应页，不清除屏幕。

依赖于其它开关的置位，这可以使得显示从低分辨率图形第一页变换到高分辨率图形的第一页，从低分辨率图形的第二页变换到高分辨率图形的第二页，或者（在正文方式下）可以使显示不改变。

200 CALL−1994

清除文本显示的第一页的上面 20 行为保留的符号@，如果你是在第一页低分辨率制图方式中，则清除制图屏幕的上面 40 行为黑色，这对正文第二页或对高分辨制图不起作用。

205 CALL−1998

清除整个文本显示的第一页为保留的符号@。如果你是在第一页低分辨率满屏制图方式中，则清除整个屏幕为黑色。这对正文第二页或对高分辨率制图不起作用。

200 CALL 62450

清除当前高分辨率屏幕为黑（BASIC 记住上次使用的屏幕，而不考虑开关的置位）。

210 CALL 62454

清除当前高分辨率屏幕为最近绘制使用的高分辨率的颜色

方式进入显示彩条图象，然后调节频道微调及饱和度，亮度，对比率旋扭，直到屏幕上的彩条满意为止。以后可把这频道作为计算机专用频道，再开机就可以不用调机了。

注意：主机显示器接口输入阻抗是 75Ω。要按电视机的要求将主机板上的"NTSC/PAL"选择插针插对。

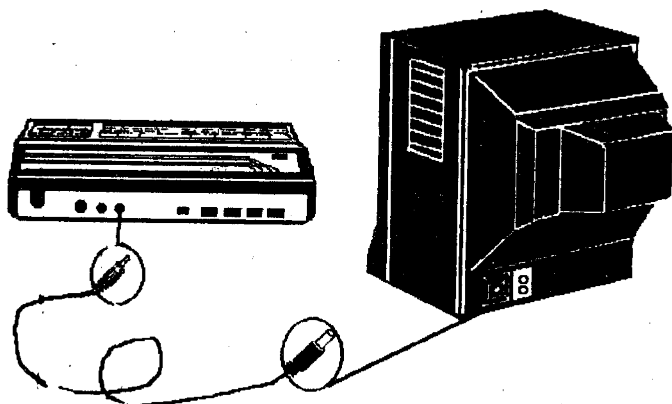


图 2—4

二、监视器连接

步聚如下：

第 1 步：将主机和监视器放在桌上，并备两个电源插座。

第 2 步：用视频电缆线一头插入主机监视器接口，一头插入监视器 IN 插座。见图 2—5

第 3 步：确认主机电源为"OFF"位置

第 4 步：将主机电源及监视电源的插头分别插入电源插

座。

第 5 步：先开监视器后开主机，屏幕将出现清晰的“ZHONG HUA XUE XI JI”字样。

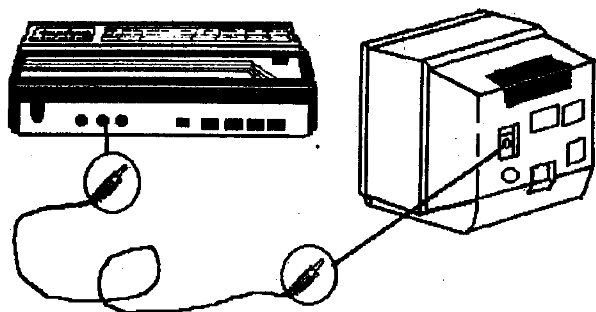


图 2—5

2.1.3 录音机的连接

第一步：确认主机电源为“OFF”位置，并再备一个电源插座（除主机、显示器电源插座）

第二步：把录音机电缆线一端（五芯插头）插入主机背面的录音机接口，必须注意：五芯插头的凹口部分在正上方。

第三步：电缆线的另一端的两个 3.5mm 插头，分别插入录音机 EAR 插孔（连接主机 IN）和 MIC 插孔（连接主机 OUT）见图 2—6

第四步:把录音机电源线插入电源插座。

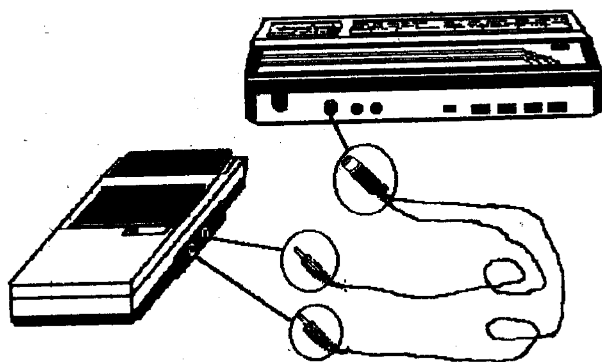


图 2—6

2.1.4 驱动器的连接

第一步:将主机电源关掉。

第二步:把磁盘驱动器电缆线(20 线扁平电缆)插座连接主机上的驱动器接口。见图 2—7

必须注意:插针与插孔必须一一对应,电缆线两端的凸出部分都朝上。只用一台驱动器时,要连接在“驱动器 1”接口上。

2.1.5 游戏杆的连接

在 CEC—2000 机壳后面有 1 个游戏杆插座,见图 2—8。这是一个两用的 15 芯游戏杆插座,可直接插入 CEC—2000 中华学习机专用游戏控制杆或任天堂游戏机用冲峰枪等。经过专用的 9 针—15 针换器可连接 CEC—1 中华学习机用游戏控制杆。

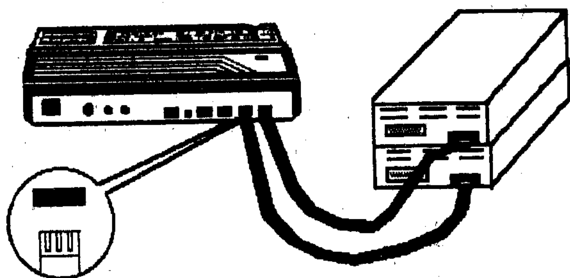


图 2—7

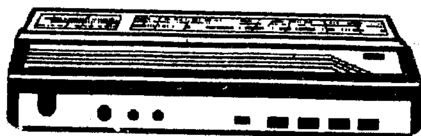


图 2-8

2.1.6 扩充卡的连接

在主机的左侧有一个 50 线的扩充槽，可插多种外设卡，扩充卡。插卡步骤如下：

第一步：将主机电源 OFF

第二步：把卡按元件面朝上，焊接面朝下的方向对准槽口往下插。必须注意：一定要平插，插平，以免错位。

扩充槽信号排列见图 2－2，如果需要重新定义槽口号，要参照图将短接帽插在标有所要槽口的号码插针上，

2.1.7　游戏机控制卡的连接。

步骤如下：

第一步：把主机电源置于“OFF”

第二步：把主机壳打开。

第三步：把游戏机控制卡，元件面朝上，焊接面朝下。把控制卡上的定位孔对准主机板上的定位孔。插牢。注意不能插错方向，不能有错针等。

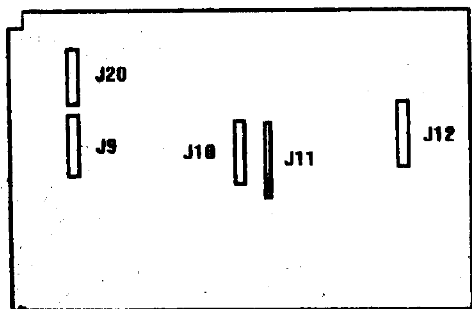


图 2—9

插入2个26针插坐后，把定位孔螺丝拧紧。（图2—9为游戏控制卡插好后状态。

2.18　内存扩展卡的连接

第一步：把学习机电源开关置于“OFF”

第二步：把机壳打开

第三步：把内存扩展卡元件面朝上，焊接面朝下，对准主机板右部的14针插座，26针插座插好。

内存卡插好后，主机内存被扩展了256K字节。

· 见图2—9

2.2　机器的启动

2.2.1　基本系统启动

首先按正确的方法连接好系统。注意将“计算机/游戏”选择开关拨至计算机一侧。请先打开显示器的电源开关，然后打开主机电源。

当一开机后，立刻可以听到“哗”的一声，这声音就告诉您，主机已准备好，同时键盘右上方的指示灯发亮。如果一切正常主机自动进入西文BASIC状态，屏幕显示如图2: 10，显示屏上有一个闪烁的小方块，它提示您下一个输入字符的显示位置，称游标。"]"是APPLE　SOFT　BASIC语言系统提示符。

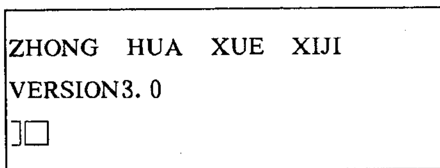


图2. 10

2.2.2 连接驱动器的系统启动

首先按正确的方法连接好系统。在开机前打开磁盘机正中央的盖子。把 DOS3.3 系统盘印字面朝上，水平而且轻轻地沿着磁盘机中间的细缝插入磁盘机内，当磁盘完全插入后，再关上磁盘机的小门。然后再开主机电源，显示器首先显示如图 2.10，磁盘机上的指示灯发亮，并且可以听到马达转动的声音，直到磁盘机灯熄灭。屏幕将显示如图 2.11 所示的版面。

此时系统已在 BASIC 系统下。如果开机前没有将 DOS3.3 软磁盘插入，系统就会直接进入 APPLE SOFTBASIC 系统。如果需要装入 DOS 系统，此时可以将 DOS 盘插入磁盘机，并键入：

]PR #6←L

于是屏幕也显示如图 2.11 所示。

```
DOS VERSION3.3              08/25/80
APPLEII  PLUS  OR ROMCARD  SYSTEM  MASTER
(LOADING INTEGER INTO LANGUAGE CARD)
]□
```

图 2.11

2.2.3 中西文切换和系统复位

一、中西文切换

当加电后屏幕显示方式为低分辨字符方式，此时若敲入中文键，系统进入汉字显示方式。屏幕显示方式为高分辨图形方式。屏幕可显示每行 17 个汉字，共 10 行，此时屏幕底部第 11 行为状态行。状态行左端显示“字母”两字，表示系统处于字符输入状态。当敲入 F2 或 F3 时，状态行左端显示“拼音”

或“区位”，表示系统处于汉字的拼音输入方式或区位输入方式。此时按 F1 可回到字母输入方式。CTRL—O 键可控制状态字符的显示与不显示，但不改变输入方式。

在中文方式下，敲入“西文”键，退出中文方式，屏幕显示回到低分辨字符方式。

除键盘方式进入汉字显示方式外，程序方式也可进入汉字显示方式。在 BASIC 程序中使用 PR #3 命令可进入汉字显示方式，使用 TEXT 和 PRINT　CHR $ (17)可退出汉字显示方式。

二、系统复位

当加电后系统运行正常，但由于操作失误或程序错误等原因使系统进入死循环或者使用者需要系统重新启动工作时，可同时按下 CTRL—RESET 键。这样系统便重新启动进入 BASIC。这种使系统复位的过程叫热启动。有时热启动不起作用，则使用者可将〈CTRL〉、〈RESET〉、〈西文〉三个键同时按下，再抬起来(西文键最后抬起)，系统将重复一次相当于打开主机电源的启动过程。有了冷启动功能可以避免因“死机”而引起电源的频繁开关。另外，也给引导磁盘操作带来方便。

2.2.4 游戏机与计算机功能切换

CEC—2000 中华学习机在插有游戏机控制卡的情况下，可以用软盘在彩电上玩任天堂游戏。具体操作步骤如下：

1. 按照前面介绍的方法分别安装好游戏机控制卡及游戏控制杆。确认安装无误后，将主机上盖装好。

2. 将主机背面的“计算机/游戏机”功能选择开关搬至“游戏机”一侧。

3. 将一台已在计算机工作方式下调整好频道的彩色电视机电源开关打开(电视与计算机之间的连接方法与计算机方

式下的连接方法相同)，音量调整至适当的位置上。

4. 将载有任天堂游戏的软盘插入软盘驱动器中，关好驱动器小门。

5. 打开主机电源，稍候片刻即可看到磁盘机工作指示灯亮，并且可以听到马达转动的声音，直到磁盘机灯熄灭，此时，即可操纵游戏杆玩游戏了。

在平时将 CEC—2000 中华学习机当作计算机使用时，只需将“计算机/游戏机”功能选择开关搬至“计算机”一侧即可，不需要将游戏机控制卡从主机中取下，下次玩游戏只要重复上述第 2 项以后的操作。

2.2.5 系统自检

CEC—2000 型中华学习机在任何时刻均可进行系统自检，方法是：同时将〈CTRL〉、〈RESET〉、〈TEST〉三个按键按下，再抬起来(TEST 键最后抬起)。系统将循环进行主 RAM，辅 RAM，ROM 及汉字库的检测，直至再次按下 CTRL－RESET 键为止。此外，在每次自检结束时，屏幕将出现彩条图案，这时按一下空格键，彩条将停留在屏幕上供调整彩电用。系统正常时的检测结果如下：(未安装辅 RAM 卡)

SELF－TEST	
MAINRAM	64K OK
AUXRAM	NULL
ROM1	32K OK
ROM2	32K OK
CECWL	256K OK

第二篇

CEC－2000 软件系统介绍

第 一 部 分　CEC-BASIC

BASIC 语言是一种简单易学、应用广泛的计算机算法语言。它的全称是 Beginer's All-purpose Symbolic Instruction Code (初学者通用符号指令代码)。它的主要特点是:

1．语言比较简单、直观、易于理解记忆，只要有一点数学基础和懂一些英语词汇，便可很快地掌握。

2．程序结构简单。由于是解释性语言，因此不用对程序进行编译。对程序的修改、检查、增删都非常容易。

3．由于是一种人机会话式语言，因此便于人和计算机间的信息交换以及程序的调试。

由于 BASIC 语言的上述特点，目前几乎所有的微型机都配有 BASIC 语言。许多个人计算机和家庭计算机都将 BASIC 语言固化在计算机上，使 BASIC 语言成为计算机的基本配置。

BASIC 语言本身随着应用的推广普及也在不断发展提高。目前 BASIC 语言已经发展出很多种类，它们大致可分为这样几种：基本 BASIC 语言，扩展 BASIC 语言和多用户分时 BASIC 语言等。它们的功能越来越强。

第一章
CEC–BASIC 语言的特点

中华学习机上的 BASIC 语言(CEC–BASIC)固化在主机电路板的只读存储器中。其功能相当于扩展 BASIC。它除了具有 Applesoft BASIC 语言的功能外还具有汉字处理等功能。当按下"中文"键后,BASIC 就可处理汉字的输入、显示和打印。

1.1 显示方式

按下中文键后,显示方式为高分辨图象方式,屏幕每行显示 17 个汉字或 34 个字母,共 10 行,第 11 行为状态行。

1.2 特殊语句

1.2.1 HGR2

进入汉字显示方式后,由于显示区使用存储器高分辨第 2 页,所以使用 HGR2 语句的效果是清屏幕,连同状态行也被清除掉。

1.2.2 HGR

使用 HGR 语句会使屏幕变为混合显示方式,即屏幕上方为图象显示,而屏幕下面有 4 行字符的显示方式,在这种显示方式下不能显示汉字。

1.2.3 FLASH

在中文方式下不提供闪烁显示方式，因此不能使用 FLASH 语句。

1.2.4 TEXT

进入汉字显示方式后，使用 TEXT 语句可回到文本显示方式。

1.2.5 TAB、VTAB 和 HTAB 语句

进入汉字显示方式前，TAB(X)和 HTAB(X)中的 X 取值为 1 ~ 255。其中 1 ~ 40 对应当前行，41 ~ 80 对应下一行，后面以此类推。而进入汉字显示方式后，X 的取值也为 1 ~ 255，但 1 ~ 34 对应当前行，35 ~ 68 对应下一行，后面类推。

进入汉字显示方式前，VTAB(X)中的 X 值为 1 ~ 24，进入汉字显示方式后，X 取值为 1 ~ 10。

1.3 增加的 BASIC 语句和功能

1.3.1 PLAY

功能：将盒式磁带机中游戏软件装入内存自动运行。

格式：PLAY

说明：先将磁带退到开始位置，在键盘上打入 PLAY，按下磁带机上的"PLAY"键，使磁带机启动，然后按下键盘上回车键(RETURN)。此时屏幕显示"WAITING …"等信息。当游戏程序装入内存后自动运行。此时，按下磁带机的 STOP 键。

1.3.2 MUSIC

功能：定义一个音阶和音长，并使扬声器发声。

格式：MUSIC X，Y

说明：X表示发声的频率，Y表示发声的时间。X、Y的取值范围在0～255之间。它们的取值与对应的音阶和音长如下表：

X值	255	228	205	192	171	152	140	128	114	102
音阶	$\underset{\cdot}{5}$	$\underset{\cdot}{6}$	$\underset{\cdot}{7}$	1	2	3	4	5	6	7

X值	95	84	75	68	62
音阶	$\dot{1}$	$\dot{2}$	$\dot{3}$	$\dot{4}$	$\dot{5}$

Y值	30	70	110	160	255
音长	1/4拍	1/2拍	1拍	2拍	4拍

例如，一个发出音阶旋律的程序可按如下方法来设计：

```
10  FOR I = 1 TO 15
20  READ X, Y
30  MUSIC X, Y
40  NEXT I
50  DATA 255, 160, 228, 160, 205, 160, 192, 160, 171,
    160, 152, 160, 140, 160, 128, 160, 114, 160, 102,
    160, 95, 160, 84, 160, 75, 160, 68, 160, 62, 160
```

1.3.3 磁带文件的访问

磁带文件的访问增加了按文件名读写文件的功能。

一、SAVE

功能：将内存中BASIC程序存入磁带。

格式：SAVE〔"文件名"〕

说明：敲入SAVE和文件名后，同时按下磁带机的

PLAY 键和 REC 键，然后按回车键。在程序装入磁带的过程中，计算机会发出两个“嘟”声，在装入完毕后，会显示 BASIC 的提示符“〕”，这时按下磁带机的 STOP 键。若不敲入文件名，也可以将文件存入磁带，但在装入内存时，就只能凭文件在磁带上的位置来装入文件了。格式中的方括号表示可选择，以后不再说明。

二、LOAD

功能：将磁带上的 BASIC 文件装入内存。

格式：LOAD 〔“文件名”〕

说明：敲入 LOAD 和文件名后，按下磁带机的 PLAY 键，然后按回车键。在程序装入内存的过程中，机器会发出一次“嘟”声，在装入完毕后，会显示被装入的文件名及 BASIC 提示符“〕”。当装入的文件不是要装入的文件时，屏幕会提示已装入的文件名和以下字样的信息行：“HAS BEEN LOADED”，并继续装入后面的文件，直到装入 LOAD 语句中所指定的文件为止。也可以不给出文件名，那么 LOAD 只将第一个文件调入内存，然后显示提示“〕。

第二章 CEC-BASIC 的变量和语句规则

2.1 变量名

在计算机程序中，变量用一些字母来表示。我们称这些

字母为变量的名字,即变量名。在 BASIC 程序中对变量名的表示作了一些规定,这些规定使变量名容易识别和处理。这些规定如下:

· 变量名在计算机内存中只分配两个字节,因此虽然变量名可以多达 238 个字符长,但只有前两个字符有效。例如:ABC 和 ABDEF 表示同一个变量名。

· 变量名的首字符必须是字母字符,随后的字符可以是字母或数字字符。

· 不能用保留字作为变量名,并且变量名中也不允许包含保留字。保留字是 BASIC 保留下来作为语句或函数用的,使用者在编程序时就不能再用作变量了。例如,用 END、FEND 做变量名都是错误的,因为 END 是保留字。

· 共有三种变量名:实型、整型和字符串型。实型变量名就是变量名本身,整型变量名就是在变量名后加上%,字符串变量名是在变量名后加 $ 。

以上介绍了变量名的表示,但每种变量名所表示的变量有哪些性质呢?请看下一节对变量本身的规定。

2.2 实型、整型和串变量

实型变量是以最多 9 位 10 进制数字的精确度表示出来的,其数域近似可达 10 的 38 次幂。即实数值域为

$$|\text{实数}| < 10^{38}$$

当$|\text{实数}| < 3 \times 10^{-39}$时,作为零处理。

BASIC 限制输出的 10 进制数的位数不能超过 9 位,但输入数值的精确度可以超过 9 位。在 BASIC 内部对数据

的处理是先把10进制数转换成2进制数进行计算，然后当你要求它显示答案时，它又转换成10进制数显示出来。由于这种转换造成的舍入误差，在你使用BASIC提供的函数时，如求平方根、除法和乘方等，有时不能给出你所希望的精确数。BASIC对计算结果的舍入按下述原则：当计算结果不足9位10进制数时给出精确值，达到或超过9位时给出9位近似值，此时对第10位作4舍5入。4舍5入的运算公式如下：

$$X = \frac{INT(X * 10^D + 0.5)}{INT(10^D + 0.5)}$$

其中INT表示截取整数，D表示小数点的位数。

整型变量表示在 -32767 至 + 32767 之间的整数值。即整数值域为：

$$|整数| \leqslant 32767$$

整型变量不允许用在FOR或DEF语句中。整型变量的最大优点是用于数组操作，数组的下标变量可以使用整型变量以节省存储器空间。

串变量表示一个字符串。与数值变量一样，串变量也能被赋以特定的值，串变量的取值范围为0～255个字符。串变量和数值变量的区别在于变量名后跟有$号。对字符串的处理，BASIC提供了一些函数。如求字符串的长度，求字符串中的子串等，这些内容将在后面的章节讨论

下面给出BASIC程序设计中使用的三种类型的变量的概述：

描　　述	变量后的符号	例　子
实型 (指数在 -38 至 + 38 之间,尾数为 9 位 10 进制)	无	A BOY
整型 (在 -32767 至 + 32767 之间)	%	B % C1 %
串型 (0 至 255 个字符)	$	D $ CITY $

所有算术运算都是以实数精确型方式来进行的,在进行运算之前,整数和整型变量值转换成实数精确型.函数 SIN, COS, ATN, TAN, SQR, LOG, EXP 和 RND 也将它们的参数转换成实数精确型,所得的结果也为实数精确型。

当一个数被转换成整数时,则取小于或等于该数的最大整数,例如:

```
10  I % = 0.999
20  PRINT  I %
RUN
0
10  A % = -.01
20  PRINT A %
RUN
-1
```

2.3 数字格式

以下是 BASIC 对输出数值格式的限定(假设 X 是作为被输出的数值):

(1) 若 $X<0$,即 X 是负数,则输出的 X 值的第一个字符是一个负号。

(2) 若 $0<|X|<999999999$,X 为整数,则 X 作为整数输出。

(3) 若 $0.01<|X|<999999999.2$,则 X 用定点表示法输出,不用指数形式表示。

(4) 若 X 不属于(2)、(3)情况,则 X 采用科学计数法表示,格式如下:

SX.XXXXXXXXEVTT

其中 S 为数的符号,正数无表示,负数用"-"表示,X 为 0~9 之间的 10 进制数字。E 表示 10 的幂。V 为指数符号,输出时正或负分别用"+"或"-"表示出来,输入时正号可省。TT 为指数值,且总是以两位数字表示。下面的例子给出 BASIC 输出各种数据的格式:

数据	输出格式
+ 1	1
-1	-1
6523	6523
-23.460	-23.46
45.72E5	45 72000
1 * 10 ∧ 20	1E+ 20
-12.34567896 * 10 ∧ 10	-1.23456789E+ 11

1000000000	1E+09
999999999	999999999

(注:其中∧表示乘方)

2.4 运算符与表达式规则

在 BASIC 中,运算符可以分为三类:

(1) 算术运算符:

+(加)、-(减)、*(乘)、/(除)、∧(乘方)

(2) 逻辑运算符:

NOT(非)、AND(与)、OR(或)、=(相等)、
< >(不等)、> <(不等)、>(大于)、
<(小于)、>=(大于等于)、=>(大于等于)、
<=(小于等于)、=<(小于等于)

(3) 串运算符:

+(串连接)

用运算符将数据、变量或函数连接起来的运算式称之为表达式。因为运算符有三种不同类型,因此也就有三种不同类型的表达式,它们分别是:

算术表达式——用算术运算符将整数、实数、算术变量、算术函数连接起来的运算式。单独形式的整数、实数、算术变量、函数,也作为算术表达式的特殊形式。

串表达式——用串运算符将字符串、串变量、串处理函数连接起来的运算式。单独形式的串、串变量也作为串表达式的特殊形式。

逻辑表达式——用逻辑运算符将任何种表达式(包括逻

辑表达式本身)连接起来的运算式。逻辑表达式的运算结果为逻辑值，逻辑值只能为 0 或 1，0 表示“假”，1 表示“真”。下面是不同类型表达式的例子：

A * 3+ (B / 2 * SIN(X))　　算术表达式

“AB”+ A $ + MID $ (A $,4,2)　　串表达式

(A= “YES”)AND(B < >9)　　逻辑表达式

NOT(5 <3)　　逻辑表达式

一个表达式中可能有多种运算符，求表达式结果时必须按照表达式的求值规则决定运算的先后顺序。表达式求值规则体现在算符的优先级上。下面按照它们执行的优先次序列出运算符。同一行中的运算符优先级相同。表达式中相同的优先级的运算符的执行是从左到右地进行。

( )

+、-、NOT 单目运算符

∧

*、/

+、-

>、<、>=、=>、<=、=<、< >、> <、=

AND

OR

2.5 语句行规则

一个语句行是以行号开始，用回车键(Return)结束。语句行号是以 0 ~ 63999 之间的无符号整数确定的。一个语句行中可包含多个语句，用冒号(:)分隔。一个语句行中最多可输入 239 个字符，其中包括行号和输入的空格。

2.6 保留字

BASIC 使用的语句，命令和函数等称为保留字，即保留给 BASIC 使用，用户不能再用保留字作为自己的变量名。保留字的具体规定见附录。

2.7 执行方式

有两种执行方式:

(1) 立即方式:不打语句行号，直接打入语句或命令，按回车键后立即被执行。

(2) 间接方式:即编程方式。每个语句要打上行号、程序执行要通过 RUN 命令。

2.8 字符串

用引号括起来的一个字符序列称为字符串。譬如:“BELL”、“APPLE”、“123”都是字符串。存放字符串的变量称为串变量。串变量名与其他变量名的区别在于变量名的最后一个字符为“$”。例如:STRING $, A $ 。BASIC 规定字符串长度范围在 0 ~ 255 之间。长度为零的字符串称为空串。一个串变量在赋值以前，它的初值为空串。例如:

```
PRINT   LEN(Q $ );  Q $;  3
0 3
```

因为 Q $ 没有被赋值，所以 Q $ 为空串，其长度 LEN(Q $) 的输出值为 0，因为 Q $ 中不包含任何字符，所以在输出

0与3之间也没有字符。

串变量在内存中是这样安排的：串变量名(二个字节)、串长度(一个字节)、串位置指针(二个字节)。这三个量存放在从 BASIC 程序末地址向上延伸的一块区域(称为变量区)中，串的字符序列存放在由 HIMEM 地址向下伸延的一块区域(称为字符串区)中。串在字符串区中的具体位置由串的位置指针确定。

BASIC 提供了一组字符串处理函数，利用这组函数和串连接运算符可以对字符串进行各种操作。字符串处理函数的讨论见后面章节。

第 三 章
CEC-BASIC 的语句和函数

3.1 基本函数

BASIC 提供的所有函数都可以在数据类型相同的表达式中使用。它们可以在立即方式中使用，也可以在间接方式中使用。

3.1.1 算术运算函数

以下函数中的 X 都可以用一个算术表达式来代替，只要表达式的结果满足 X 的条件即可。

SIN (X)——等于 X(弧度值，以下二函数同)的正弦值 sinX。

COS (X)——等于X的余弦值cosX。

TAN (X)——等于X的正切值tgX。

ATN (X)——等于X的反正切值arctgX $\left(-\frac{\pi}{2}\sim+\frac{\pi}{2}\right)$。

ABS (X)——等于X的绝对值$|X|$。

SQR (X)——等于X的平方根$\sqrt{X}$ $(X>0)$。

EXP (X)——等于X的指数值e^X $(e=2.718289)$。

LOG (X)——等于X的自然对数值lg(X) $(X>0)$。

INT (X)——等于最邻近X而又不大于X的最大整数值。例INT(3.999)=3; INT(−7.25)=−8。

SGN (X)——等于X的符号值,规定如下:

$$SGN(X)=\begin{cases}1 & (X>0)\\0 & (X=0)\\-1 & (X<0)\end{cases}$$

RND (X)——等于一个随机数,此数视X值而有下述区别:

(1) X >0,产生0 ~ 0.999999999间的一随机数。

(2) X=0,产生与上次使用此函数时产生的相同值。

(3) X <0,产生一个与X值一一对应的数。

下面即为一段打印RND(X)数值的程序:

```
10  PRINT "RND(1)=";RND(1)
20  PRINT "RND(1)=";RND(1)
30  PRINT "RND(0)=";RND(0)
40  PRINT "RND(0)=";RND(0)
50  FOR I=1 TO 2
```

```
60  PRINT "RND(-1)=";RND(-1);SPC(5);
70  PRINT "RND(-2)=";RND(-2)
80  NEXT
]RUN
RND(1)= .273385388
RND(1)= .621966945
RND(0)= .621966945
RND(0)= .621966945
RND(-1)=2.99196472E-08
RND(-2)=2.99205567E-08
RND(-1)=2.99196472E-08
RND(-2)=2.99205567E-08
```

该程序的第 10 行、20 行都是调用 RND(1)函数，从运行结果看，两个调用 RND(1)的函数值不同，说明 RND(X)函数的随机性。第 20 行与 30 行、40 行的运算结果相同，说明 RND(0)不会产生新的数值，只是保留上一次 RND(X)值。从 50 到 80 行的运行结果看，说明对于每一个小于零的 X 值，RND(X)都有与之对应的固定函数值，函数值不随机变化。当你在程序设计中需要使用一组没有规律的数字作为运算参数时，可以使用随机函数来产生这组数据。我们称这些没有变化规律的数字为随机数。

3.1.2 派生函数

下面的函数，虽然不是 BASIC 直接提供的函数，但是，它们能够用现有的 BASIC 函数来计算，并且可以通过使用 DEFFN(自定义函数)的功能很容易地实现。

正割：SEC(X)=1/COS(X)

余割：CSC(X)=1/SIN(X)

余切: COT(X) =1 / TAN(X)

反正弦: ARCSIN(X) =ATN(X / SQR(−X * X+ 1))

反余弦: ARCCOS(X) =−ATN(X / SQR(−X * X+ 1)
+ 1.5708

反正割: ARCSEC(X) =ATN(SQR(X * X−1))
+ (SGN(X)−1) * 1.5708

反余割: ARCCSC(X) =ATN(1 / SQR(X * X−1))
+ (SGN(X)−1) * 1.5708

反余切: ARCCOT(X) =−ATN(X)+ 1.5708

双曲正弦: SINE(X) =(EXP(X)−EXP(−X)) / 2

双曲余弦: COSH(X) =(EXP(X)+ EXP(−X)) / 2

双曲正切: TANH(X) =−EXP(X) / (EXP(X)
+ EXP(−X)) * 2+ 1

双曲正割: SECH(X) =2 / (EXP(X)+ EXP(−X))

双曲余割: CSCH(X) =2 / (EXP(X)−EXP(−X))

双曲余切: COTH(X) =EXP(−X) / (EXP(X)
−EXP(−X)) * 2+ 1

反双曲正弦: ARGSINH(X)
=LOG(X+ SQR(X * X+ 1))

反双曲余弦: ARGCOSH(X)
=LOG(X+ SQR(X * X−1))

反双曲正切: ARGTANH(X)
=LOG((1+ X) / (1−X)) / 2

反双曲正割: ARGSECH(X)
=LOG(ISQR(−X * X+ 1)+ 1) / X

反双曲余割: ARGCH(X) =LOG(SGN(X)
* SQR(X * X+ 1)+ 1) / X

反双曲余切: ARGCOTH(X)

=LOG((X+1)/(X-1)/2)

A MOD B:　MOD(A)=INT((A/B-INT(A/B))

*B+0.05)*SGN(A/B)

3.1.3 字符串处理函数

一、LEFT $ (A $,n)

该函数用来取一个字符串的左子串(最左边的若干字符)。A $ 表示被处理的母串,n 值确定子串长度。n 为实数时,将自动转换为整数。要求 1 < n < 255,否则会出现"ILLEGAL QUANTITY ERROR"错误。例如,假定我们给 A $ 赋予字符串"GOOD MORNING",现在我们只想打印出"GOOD",我们可以这样做:

```
10  A $ ="GOOD MORNING"
20  PRINT LEFT $ (A $ ,4)
RUN
GOOD
```

二、RIGHT $ (A $,n)

该函数用来取字符串的右子串(最右边的若干字符)。其它说明与 LEFT $ 函数相同。对于 LEFT $ 函数中的例子,如果我们只想打印出"MORNING",我们可以使用 RIGHT $ 函数。

三、MID $ (A $,n;m)

MID $ 函数的名字取自英文 MIDDLE(中间)。顾名思义,该函数可以取一个字符串中间部分的子串。A $ 表示被处理的母串。n 表示从母串中摘取子串的第一个字符位置,m 表示子串的长度。若 m 被省略,则从 n 位置开始,一直到母串的最右端一个字符构成子串,其效果与 RIGHT $ 函数相

同。

n 和 m 的取值范围分别为 1 ≤ n ≤ 255，1 ≤ m ≤ 255，若 n, m 超出值域，也会出现："ILLEGAL QUANTITY ERROR"错误。若 n 大于母串长度，则该函数返回一个空串。例如：

```
]PRINT MID $ ("ABCD", 5, 10)
```

你会发现屏幕上没有结果产生，而实际上产生一个空串。因为"ABCD"长度为 4，而 n=5，所以 MID $ 函数返回一个空串

如果我们用 LEN(A $)表示 A $ 中字符串的长度，则当 m >LEN(A $)−(n−1)时，m 多出的量被忽略，系统自动将 m 置为 LEN(A $)−(n−1)的值。例如：

```
]PRINT MID $ ("ABCD", 3, 5)
CD
```

因为 n=3，所以最长子串长度为 2，虽然在函数中子串长度 m=5，但是实际上取出的子串长度为 2。

以上所讨论的三个函数共同特点是：它们的函数值都是字符串。

四、LEN(A $)

此函数用来求一个字符串的长度，即计算出字符串中包含的字符总数，它的返回值是一个整数而不是字符串。例如：

```
10  A $ ="BASIC"
20  PRINT LEN(A $ )
RUN
```

以上这4个字符串处理函数中的A $可以用串表达式代替;n或m也可以用算术表达式代替。下面是利用字符串处理函数编写的一个小程序,通过该程序介绍字符处理的一般方法。

```
10  A $ ="China Education Computer"
20  B $ =LEFT $ (A $ ,5)
30  C $ =MID $ (A $ ,7,9):D $
     =RIGHT $ (A $ ,8)
40  PRINT B $ ,C $ ,D $
50  PRINT "LEN(B $ )=";LEN(B $ ),"LEN(C $ )
    =";LEN(C $ ),"LEN(D $ )=";LEN(D $ )
60  PRINT B $ + " "+ C $ + " " + D $
RUN
China              Education         Computer
LEN(B $ )=5     LEN(C $ )=9     LEN(D $ )=8
China Education Computer
```

3.1.4 类型转换函数

本节中介绍的函数作用是做数据类型的转换,把算术型数据转换成字符型或字符型数据转换成算术型。

一、STR $ (算术表达式)

该函数将算术表达式的值转换成字符型的数值,例如:

```
]PRINT STR $ (56 * 10 ∧ 12)
   5.6E+ 13
]PRINT 30+ STR $ (56 * 10 ∧ 12)
   ?TYPE MISMATCH ERROR
```

执行第二条命令时发生类型失配错误,这是因为用STR $ (X)函数后5.6E+ 13变成字符类型,而30是算术类

型，字符型与算术型的数据是无法做加法运算的

二、VAL(串表达式)

此函数将一字符型数据转换成算术型。假定串表达式本身或它的前一部分为一数字形式，则此函数值即为此数字；假定串表达式开头部分非数字形式，则函数值为0。串表达式的长度不能超过255；VAL函数的绝对值不能大于1E38，否则将发生错误。例如：

```
]PRINT VAL(".456")
 .456
]PRINT VAL("+ 123AD")
 123
]PRINT VAL("1E+ 10")
 1E+ 10
]PRINT VAL("E23")
 0
]PRINT VAL("20+ 30")
 20
]PRINT VAL("0E8A")
 0
```

三、CHR $ (算术表达式)

此函数返回一个与算术表达式值相对应的ASCII字符。若算术表达式为一实数，将自动转换成整数后再进行函数转换运算。算术表达式的值必须在0～255之间，否则出现"ILLEGAL QUANTITY ERROR"错误。例如：

```
]PRINT CHR $ (65),CHR $ (30)
   A           0
```

]PRINT CHR $ (20+41)

=

四、ASC(串表达式)

此函数返回串表达式值的第一个字符的 ASCII 码。串表达式值的长度不能超过 255,否则出现错误。例如:

]PRINT ASC("WER"),ASC("A")

87　　　65

"W"的 ASCII 码为 87,"A"的 ASCII 码为 65。

]PRINT ASC(STR $ ("23")),ASC(CHR $ (65))

50　　　65

ASC(STR $ (23))等价于 ASC("23"),因为"2"的 ASCII 码为 50,所以第一项输出值为 50。第二项中的 CHR $ (65)相当于"A",所以输出 A 的 ASCII 码 65。

3.2 基本语句

3.2.1 LET

功能:将等号右边的表达式值赋给左边的变量。LET 可省略。

格式:〔LET〕算术变量〔下标〕=算术表达式

〔LET〕串变量〔下标〕=串表达式

说明:算术变量在赋值之前初值为 0,串变量初值为空。表达式类型必须与被赋值变量的类型一致。算术型表达式值不能赋给串变量,反之亦然。实型变量与整型变量之间赋值能自动进行转换和取舍。将实数型值赋给整型变量时,自动将实数取整,并以整数格式存放到该变量中;将整型值赋给实型变量时,自动转换成实型数格式存放在变量中。赋值语句

LET 的例子如下:

LET A =138+ 56.72 * SIN(5)

B(3)% =20+ 1

CC $ ="HAPPY"+ "TO YOU"

3.2.2 GOTO

功能:执行该语句时,计算机转向指定的行号去执行。

格式:GOTO 行号

3.2.3 条件语句

功能:根据表达式的值确定程序转向。

格式:IF 表达式 THEN 语句:语句…语句

IF 表达式 THEN 行号

IF 表达式 GOTO 行号

说明:表达式可以是算术表达式、字符串表达式或逻辑表达式。执行该语句时,如果表达式的值非零(即条件为负),则执行“THEN”后面的语句,否则继续执行下一程序行。表达式是否为零的确定方法如下:

- 当表达式的绝对值小于等于 2.93873E-39 时,其值为零。
- 字符串表达式的结果为空串时,其值为零。
- 逻辑表达式中的关系不满足时,其值为零。

3.2.4 循环语句

功能:循环执行 FOR 与 NEXT 之间的程序行。

格式:FOR 实型算术变量 =算术表达式 1 TO 算术表达式 2〔STEP 算术表达式 3〕

程序行

…

NEXT〔算术变量〕〔{,算术变量}〕

说明：循环可以嵌套，但至多可以嵌套十层。不同级的循环不能相互交叉。循环变量一定要用实型变量。NEXT 语句后的循环变量可以省略。当多重循环语句嵌套，且有同一出口时，可以共用一个 NEXT，但是在 NEXT 之后要将多个循环变量按照嵌套从内到外的顺序一一列出，循环变量之间用逗号分隔。在循环体内可以用 GOTO 语句跳出循环，但不允许用 GOTO 语句从循环体外直接转入循环体内。

在立即执行方式下使用时，要求 FOR 与 NEXT 必须在一个程序行中（即它们之间不能有回车），并且该程序行的字符个数不能超过 239 个。

3.2.5 READ 和 DATA

功能：将 DATA 语句所给的数据序列中的数据依次赋给 READ 语句中的各个变量。

格式：READ〔变量名〕〔{，变量名}〕

DATA 数据〔{数据}〕

说明：DATA 语句中的数据可以是算术型数值，也可以是字符型的串，但不能是表达式。程序中可以有多个 DATA 语句，DATA 语句可以放在程序中的任何位置，可以在 READ 语句之前，也可以在 READ 语句之后。所有 DATA 语句的数据序列构成一张数据表。在这张数据表中有一个数据指针，程序运行之前，该指针指向数据表中的第一个数据，每执行一次 READ 语句，“READ”之后的各个变量就被赋值，数据表中的数据指针就被相应向右移动几个位置。

例：

```
10  DATA 1,-,2,-,3
20  DATA -,4,--,5,-,6,-
```

```
30  FOR I=1 TO 6: READ NUM,ST $
40  PRINT NUM;ST $;
50  NEXT
]RUN
1 — 2 — 3 — 4 — 5 — 6 —
```

在使用 READ 与 DATA 语句时要注意几点：

• 每次执行 READ 语句时，变量类型要与相对应的 DATA 中的数据类型一致，否则出现：

"?SYNTAX ERROR IN 行号"

• DATA 语句中的字符型数据，可以用引号将其括住表示，也可以不用，但对于数字型，其类型取决于 READ 语句中相应变量的类型。

• 若 READ 语句所读数据超过 DATA 数据表中的最后一个数据，则出现："OUT OF DATA ERROR IN 行号"错误。

3.2.6 RESTORE

功能：将 DATA 数据表中的指针指向第一个数据，使 DATA 的数据可以重新使用。

格式：RESTORE

3.2.7 GOSUB

功能：转到以指定行号为首行的 BASIC 子程序上去执行，同时将 GOSUB 语句之后的语句地址"推入"堆栈，以便子程序执行结束后返回该地址继续执行。

格式：GOSUB 行号

说明：RETURN 语句是子程序结束标志，它将堆栈中保存的返回地址"弹出"，使计算机按照该地址继续执行。子程

序嵌套最多25层。

3.2.8 多向转移和多向转子

功能:根据n的值,选择第n个行号,使计算机转向该行号执行。

格式:ON n GOTO 行号1,行号2,…行号i

ON n GOSUB 行号1,行号2,…行号i

说明:其中n为正整数,并满足0<n<255。n也可以用算术表达式代替,如果n的值为实数,则自动转换为整数;如果n值大于最大行号或n=0,则该语句不执行,跳到下一语句执行。如果n<0,则出现:"?ILLEGAL QUANTITY ERROR IN 行号"错误。

3.3 输入输出语句

3.3.1 INPUT

功能:从当前输入设备上接收各种类型的数据,依次赋给其后的变量。

格式:INPUT〔串;〕 变量〔{,变量}〕

说明:大括号{ }内的部分是可重复出现的部分,在INPUT之后可以紧跟一个提示字符串,该字符串后必须是分号(;)。当执行到INPUT语句时,如果INPUT之后有提示字符串,则在屏幕上显示该字符串,然后等待输入。如果在INPUT之后没有提示字符串,则在屏幕上显示一个问号,然后等待输入。输入多个数据时,用逗号分隔,用回车键结束数据输入。如果输入的数据少于"INPUT"之后的变量个数,屏幕上会提示"??"二个问号,继续等待输入。如果输入的数据多于"INPUT"之后的变量个数时,屏幕上提示:"EXTRA

IGNORED",但不影响程序执行,只是将多余的输入数据忽略。CTRL-C/ 可以中断 INPUT 语句,但是要求 CTRL-C 必须是 INPUT 接收的第一个数据。

用 INPUT 语句输入多个数据时,也可以用 RETURN 作为数据间的分隔。

输入的数据类型要与相对应的变量类型一致,否则给出提示"?REENTER",要求重新输入。

用 INPUT 输入字符串型数据时,字符串中不能有逗号,可以有分号、引号及其它特殊符号,因为逗号在此被识别为分隔符。

3.3.2 GET

功能:从当前输入设备上接收一个字符或一位数字赋给变量。

格式:GET 变量名

说明:在执行 GET 时,屏幕上没有任何提示信息,输入的字符也不显示在屏幕上。输入数据之后不用按回车键。

对于"GET 算术变量"这种格式,如果输入 +、-、·、,、:、那么实际赋给变量的值都为零;输入除上述字符外的其他非数字字符时,将显示"?SYNTAX ERROR"表示句法错,且程序被停止执行。由于这些原因,"GET"之后最好不用算术变量,否则在输入时限制较多,容易发生错误,一旦按下一个字母或 RETURN,则使程序运行中途而废。如果在"GET"之后用一个串变量,就可以避免上述问题,当需要输入一个算术型数字时,可以先以串变量接收,之后再用 VAL(X)函数将其转换成算术型数字。

另外,用 GET 语句输入数据时,不论输入多少个字符或多少位数字,实际接收的只是第一个字符或数字。

3.3.3 PRINT

功能:将各种数据或表达式值输出到屏幕或指定的输出设备(如打印机)上。

格式:PRINT〔{打印格式}〕〔表达式〕

说明:其中的打印格式由表达式后接分隔符(,或;)组成。可用“?”代替“PRINT”语句。用分号作分隔符时,前后表达式的结果是紧接着输出;用逗号作分隔符时,则按制表域输出表达式结果,即每个表达式的结果占16个字符位置。第一个制表域由最左边16个字符位置组成(位置1至16),第二个制表域占有随后的16个位置(17至32),如果第一个制表域的第16个位置也有输出的字符,则第二个制表域就放到第三个制表域,否则两个数据首尾相接会不易辨认。第三个制表域为剩下的8个字符位置(33至40)。若第三个制表域的输出字符多于7个,则第三制表域会放到下一行的第一制表域。若输出字符个数超过16,则自动占用下一个制表域的位置。

3.3.4 SPC

功能:以上一次输出的最后一个字符位置为基准,跳过若干空格,空格数由算术表达式值确定。

格式:SPC(算术表达式)

说明:SPC是一个函数,它必须放在PRINT后使用。算术表达式的值必须在0~255之间。

3.3.5 VTAB

功能:将光标移到由算术表达式的值确定的行上。

格式:VTAB(算术表达式)

说明:VTAB只能使光标上下移动,不能左右移动。在西文状态下,算术表达式值必须在1~24之间。在中文状态

下，在 1 ~ 10 之间。

3.3.6 HTAB

功能：将光标移到由算术表达式的值确定的列上。

格式：HTAB(算术表达式)

说明：在西文状态下，算术表达式值必须在 1 ~ 255 之间。以光标所在行的 1 到 40 列为当前行，41 ~ 80 列为下一行；在中文状态下，算术表达式的值也在 1 ~ 255 之间。而 1 ~ 34 为当前行，35 ~ 68 为下一行。

3.3.7 TAB

功能：在当前行上以最左列(第一列)为基准，将光标转到指定的列位置，列位置由算术表达式值确定。

格式：TAB(算术表达式)

说明：此语句与 HTAB 的区别是 TAB 只能使光标右移，而不能向左回退。此命令必须在 PRINT 之后使用。在中、西文两种状态下，算术表达式的值均应在 0 ~ 255 之间。

3.3.8 POS

功能：送回当前光标所在的列位置。

格式：POS(表达式)

说明：表达式的值没有意义，可简单地放 0 或 1。POS(X)的值为 0 时，认为是最左列。

3.4 显示方式的控制语句

3.4.1 NORMAL

功能：置显示器为正常显示方式，即“黑底白字”。

格式：NORMAL

3.4.2 INVERSE

功能：置显示器为反相显示方式，即"白底黑字"。

格式：INVERSE

3.4.3 FLASH

功能：置显示器为闪烁方式。

格式：FLASH

说明：中文状态下不支持此语句。

3.5 定义语句

3.5.1 DIM

功能：定义一个数据矩阵或数组，并为该数组在内存中开辟数据区域。数组中的每一个元素作为一个变量使用。

格式：DIM 变量名(下标)〔{,变量名(下标)}〕

说明：数组维数最多可达 88 维，然而实际使用该语句时，维数的多少是受内存量限制的，往往达不到 88 维。下例为一个二维数组，它的每一个元素都是一个变量。每维的标量为 2.

```
                    A(0,0)A(0,1)A(0,2)
DIM A(2,2)──→   A(1,0)A(1,1)A(1,2)
                    A(2,0)A(2,1)A(2,2)
```

数组中每一维的最小下标量为零。例如 DIM A(3)实际有四个元素：A(0),A(1),A(2),A(3)。每一维的标量可达 32767，但由于内存量限制，往往达不到 32767。

数组也有类型之分，定义数组类型的原则同变量名类型原则一致，在数组名之后附加类型符"$"或"%"，数组中每一元素的类型也就随之确定了。例：DIM A%(5)，STR$(10)。

如果没有用 DIM 语句定义，直接使用数组的元素变量名，在这种情况下系统并不认为出现错误，而是自动将其定义成每维标量为 10 的数组。

一个矩阵只能定义一次(包括没有用 DIM 语句定义而交由系统自动定义的情况)，否则会出现“?REDIM ARRAY ERROR”错误。

当执行 RUN 或 CLEAR 命令时，内存中的数组元素都被清除。

3.5.2 自定义函数

功能：允许用户在程序中自己定义一个函数。

格式：DEF FN 变量名(实型算术变量) = 算术表达式

说明：由 DEF 语句定义的函数—— FN 变量名(X)的使用就如同 SIN(X)，COS(X)一样，X 作为该函数的自变量，X 也可以用算术表达式代替。FN 变量名(X)的函数值是将 X 代入 DEF 语句中的表达式运算之后的结果。

例如：用自定义函数的方式计算各种 X 的 $(X^2+X)/2\times10$ 的值。用 CTRL−C↙ 结束运行。

```
10  DEF FN P(X)=(X∧2+X)/2
20  INPUT "INPUT X=";X
30  PRINT X, FN P(X)*10
40  GOTO 20
]RUN
INPUT X=3
3          60
INPUT X=42
42         9030.00001
```

在使用自定义函数时要注意以下几点：

· 自定义函数只能是算术函数，不允许是串函数。

· DEF 语句必须在所有调用函数的语句行之前，即“先定义后调用”。

· 自定义函数只能有一个自变量，但是对该自变量在算术表达式中出现的次数不限。

· 自定义函数中的自变量可以是“虚”元，即在表达式中可以不出现该自变量名，例如：DEF FN P(A) =Z ∧ l+ 3 ∧ I 是允许的。

· 自定义函数中的自变量必须用实型变量表示，如果用整型变量则会出现："SYNTAX ERROR IN 行号"错误。

· 对于同一个自定义函数名可以多次重新定义，例如：

```
10   DEF FN AB(I) =SIN(I)
15   PRINT FN AB(0)
20   DEF FN AB(I) =COS(I)
25   PRINT FN AB(0)
] RUN
0
1
```

3.6 其它语句与实用函数

3.6.1 PEEK 和 POKE

功能：使用户可以直接读写内存单元的值。

格式：PEEK (X)

　　　POKE X，Y

说明：PEEK 函数返回由 X 值表示的内存地址单元中的内容，函数值用 10 进制数表示。X 值应在 −65535 和

65535 之间。X 也可以用算术表达式代替。

POKE 语句的作用是将 10 进制数 Y 存储到 X 值表示的内存地址单元中。Y 值必须在 0 ~ 255 之间,X 值必须在 -65535 和 65535 之间。

3.6.2 LOMEM 和 HIMEM

功能:限制 BASIC 程序工作区域的上、下限地址。目的是防止 BASIC 的数据破坏高分辨图象显示区域或机器语言子程序。

格式:HIMEM: X

LOMEM: X

说明:在引导 DOS 系统之后,HIMEM 值自动设置为 $ 9600,LOMEM 值一般由"HELLO"程序的大小决定,即指向内存中 BASIC 程序的尾部。若不装入 DOS,则 HIMEM 值自动设置为 $ C000,LOMEM 设置为 $ 0804。下图为装入 DOS 后在 BASIC 状态下的内存分配图。

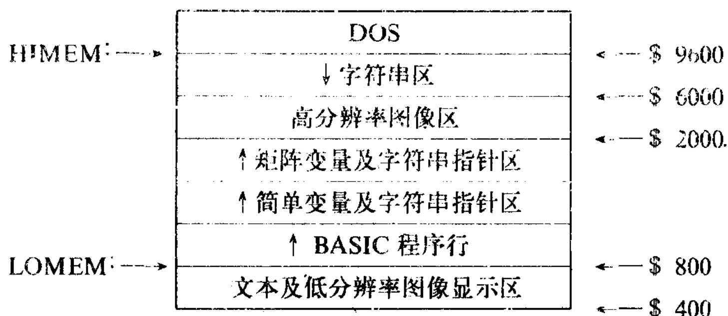


图 1.3.1 BASIC 状态下的内存分配图

从内存分配图上我们可以看到,BASIC 的字符串正是在高分

辨率图象区之上，因为字符串区是向下延伸的，当它大到必须侵占高分辨图象区时，高分辨图象区就会被破坏。再譬如，BASIC 程序中要调用机器语言子程序，将机器语言子程序放在 $ 801 ~ $ 9600 之间的某一区域中。这时该子程序很容易被 BASIC 程序的数据破坏。因为这个区域是系统自动设置的 BASIC 工作区，随时都有可能占用这个区中的任何一个位置。LOMEM 和 HIMEM 给用户提供了自己设置 BASIC 工作区域的手段。如果要用高分辨率图象方式作图，执行 HIMEM 8191 命令就可以实现对高分辨显示区进行保护的目的。也可以通过将机器语言子程序放在 HIMEM 设置的地址之上的空闲区中的办法，使之免遭破坏。

由于 CEC-BASIC 在中文状态下使用 $ 9200 到 $ 9600 作为屏幕映射区，所以进入中文状态后 HIMEM 自动设置到 $ 9200。

内存中的 BASIC 程序的第一条语句都是从 $ 801 地址开始向上存放的，程序最末尾地址加 4 就表示当前“LOMEM 地址”。“LOMEM 地址”是随程序行的增加、删除而自动调整的。

LOMEM 和 HIMEM 所置的地址值必须在 -65535 ~ 65535 之间的 10 进制数，该地址若为实数将自动转换成整数。

LOMEM 的当前地址值存放在 106 与 105 单元中(即 $ 6A、$ 69 单元)；HIMEM 的当前地址值存放在 116 和 115 两个单元中(即 $ 74、$ 73 单元)。可以执行以下命令得到它们的当前值。

```
?PEEK(106) * 256+ PEEK(105)↙
?PEEK(116) * 256+ PEEK(115)↙
```

3.6.3 FRE(X)函数

功能：释放数据区中的无用单元，返回可用变量的内存量，即余下的简单变量空间和数组变量空间。

格式：FRE(X)

说明：X 可以用表达式代替。但 X 实际上是一个"虚"元，设为零即可。

在程序的运行期间，当一个串的内容被修改时，BASIC 并不清除旧的字符串内容，而是给新的串内容开辟内存单元。其结果是大量的废弃字符慢慢从 HIMEM 向下填补。如果你正在使用机器语言程序或高分辨率显示方式，你的程序区或高分辨率图象区有可能被字符串变量破坏。在这种情况下，定期使用 X=FRE(0)的语句可防止被破坏的结果产生。

3.6.4 SPEED

功能：改变计算机到 I/O 设备的字符传送速度。表达式之值确定速度的大小，最低速度为零，最高速度为 255。

格式：SPEED=算术表达式

3.6.5 CALL

功能：调用机器语言子程序。

格式：CALL　算术表达式

说明：算术表达式值在 -65535 ~ 65535 之间。算术表达式的值即为机器语言子程序的首地址，机器语言子程序的返回指令 RTS 可使程序的执行回到 BASIC 程序并执行 CALL 后面的语句。

3.6.6 TRACE

功能：跟踪程序的执行，将执行过的行号显示在屏幕上。

格式：TRACE

3.6.7 NOTRACE

功能:取消 TRACE 语句的作用,即停止跟踪程序的执行。

格式:NOTRACE

说明:只有 NOTRACE 才能取消 TRACE 的作用。RUN、CLEAR、NEW、DEL 与 RESET 都不能解除 TRACE 的作用。

3.6.8 WAIT

功能:使得程序在执行过程中处于等待状态,直到某个指定单元的值变化成满足一定要求时才能继续执行。

格式:WAIT X,Y

WAIT X,Z,Y

说明:X 值表示一个存储单元地址,它必须在 -65535 和 65535 之间。Y 与 Z 分别表示两个 0 ~ 255 之间的 10 进制数,执行 WAIT 命令时,它们被转换成 2 进制数。Z 可以选择使用。X,Y,Z 可用算术表达式代替。

在第一种格式中,程序先读 X 单元,所读出的内容和 Y 值相与,结果若为零则重复这个过程,结果非零,程序便向下继续执行。

在第二种格式中,程序读 X 单元后所得值与 Z 值先异或,然后再和 Y 值相与。结果为零,则重复这个过程;结果非零,则向下继续执行。

3.6.9 POP

功能:与 RETURN 语句相似。差别在于当执行 RETURN 语句时,堆栈顶部的地址被弹出,使计算机按照该地址返回继续执行;当 POP 被执行时,栈顶地址也被弹出,但是计算机不返回到这个地址执行,而是继续 POP 的下一个语

句。因此在 POP 之后执行一个 GOTO 语句就可以使计算机转向任何地方去执行。

3.6.10 PDL 与游戏控制器的使用

一、PDL(n)函数

功能：此函数可用来读取 n 号（n=0 ~ 3）摇杆位置量的编码值(编码值在 0 ~ 255)。

格式：PDL(n)

说明：下例为读 1 号摇杆位置并进行输出的程序：

```
10  G=PDL(1)
20  PRINT TAB(15); G: FOR I=1 TO 100: NEXT
30  GOTO 10
```

如果用连续的 PDL 指令来读入二个游戏控制器的数据，那么第二个游戏控制器上的读数，可能受到第一个控制器上的读数的影响。要获得更精确的读数，可以在 PDL 指令之间放几个程序行或在 PDL 指令之间置一个短的延迟程序(FOR I=1 TO 10 : NEXT I)。

二、读入开关量

已知三个开关量的地址为 $ C061 ~ $ C063(或者 $ C069 ~ $ C06B)，此即相当于 −16287 ~ −16285(或者 −16279 ~ −16277)。因此下面的语句可读入三个开关量：

X=PEEK(−16287+ n)　(n=0 ~ 2)

3.6.11 ONERR GOTO 及 RESUME

功能：出错后转处理程序。

格式：ONERR GOTO 行号

　　　RESUME

说明：在程序执行过程中，如果发生错误，计算机会转到由本语句行号指定的“纠错处理”子程序入口执行。

RESUME 是“纠错处理”子程序的返回语句。内存 216 单元为出错标志单元，若该单元的 B7 位为 1，则表示有错误发生。发生错误后，错误类型码存放在 222 单元，下表为错误类型编码表：

类型码	出错情况
0	NEXT 与 FOR 不匹配
16	句法错
22	RETURN 与 GOSUB 不匹配
42	缺少数据
53	非法量
69	溢出
77	缺少内存
92	未定义语句
107	不适当的下标
120	重定义数组
133	除数为零
163	类型不匹配
176	串太长
191	形式太复杂
224	未定义函数
254	响应 INPUT 语句时输入不适当
255	用 CTRL-C 中断纠错程序

下面的程序说明 ONERR GOTO 语句的用法，及“纠错”子程序的编写方法。此程序处理了“非法量”与“除数为零”两种类型的错误。为了帮助理解，将程序的执行过程用 TRACE 命令记录下并显示出来。

```
 10  ONERR GOTO 100
 20  T=-5
 30  A$ =LEFT$ ("Education Computer", T)
 40  PRINT A$
 50  S=300
 60  ONERR GOTO 100
 70  A(5) =S / A(0)
 80  PRINT "A(5) =";A(5)
 90  END
100  POKE 216,0
110  PRINT "PEEK(222) =";PEEK(222)
120  IF PEEK(222) =53 THEN T=9
130  IF PEEK(222) =133 THEN A(0) =5
140  RESUME
]RUN
PEEK(222) =53
Education
PEEK(222) =133
A(5) =60
]TRACE
]RUN
#10  #20  #30  #100 #110 PEEK(222) =53
#120 #130 #140 #30  #40  Education
#50  #60  #70  #100 #110 PEEK(222) =133
#120 #130 #140 #70  #80  A(5) =60
#90
```

3.6.12 USR(X)函数

功能:为用户提供的利用机器语言子程序对算术表达式 X 的值进行加工处理的函数。

格式:USR(X)

说明:此函数的执行过程如下:

(1) 计算出表达式 X 的值,将其放在浮点累加器(即 $ 9D ~ $ A3 单元)中。

(2) 转到 $ 0A 单元去执行。通常在 $ 0A ~ $ 0C 单元中存放一条 JMPXXXX 指令,其中 XXXX 是用户自己所设置的处理 $ 9D ~ $ A3 单元中 X 值的子程序入口。

(3) 在机器语言程序执行完毕(即遇到 RTS)后,该函数所得到的值将放在浮点累加器中。

3.7 低分辨率彩色作图

3.7.1 GR

该语句的作用是将屏幕上部置成 40 × 40 可显示的小方块,作为低分辨率图象显示区,将屏幕下部四行作为文本显示区,即置成低分辨图象与文本的混合显示方式。

执行 GR 后,屏幕上图象显示区被置为黑色,因此 GR 命令有清除低分辨率图象画面的作用。同时原低分辨率图象区中的图象也被破坏。

从低分辨图象与文字混合显示转到纯低分辨图象显示,可用 POKE 或 PEEK 语句访问地址 -16302(相当于 $ C052)。再回到混合方式时可访问地址 -16301(相当于 $ C053)。这两个地址是控制屏幕显示方式的软开关。用"GR"命令返回混合显示方式也可以,但屏幕被清除。

3.7.2 COLOR

功能：置低分辨率图形的颜色。

格式：COLOR =n

说明：如果 n 为实数，则将其自动转换为整数。n 取值范围在 0 ~ 255 之间，大于等于 16 的 n 自动以 16 为模处理。如果不执行该语句，系统自动将图形置为黑色。n 值与颜色的对应关系如下表：

n 值	颜色
0	黑
1	红
2	深蓝
3	紫
4	深绿
5	灰 1
6	中蓝
7	浅蓝
8	棕
9	橙
10	灰 2
11	粉红
12	浅绿
13	黄
14	绿蓝
15	白

3.7.3 SCRN 函数

功能：取得低分辨率图形方式下某一点的颜色值。

格式：SCRN(X,Y)

说明: (X, Y)为所取颜色值的点的坐标。一般情况下 X 在 0 ~ 39 之间, Y 在 0 ~ 47 之间取值。

3.7.4 PLOT

功能: 在低分辨率图形方式下绘出一个点。

格式: PLOT X, Y

说明: (X, Y)为所作点的坐标。点的颜色由 COLOR 语句设定。

3.7.5 HLIN

功能: 在低分辨率图形方式下, 绘出一条水平线。

格式: HLIN X1, X2 AT Y

说明: (X1, Y)为水平线的起点, (X2, Y)为水平线的终点。

3.7.6 VLIN

功能: 在低分辨率图形方式下, 绘出一条垂直线。

格式: VLIN Y1, Y2 AT X

说明: (X , Y1)为垂直线的起点, (X, Y2)为垂直线的终点。

3.7.7 TEXT

功能: 由低分辨图形方式或高分辨图形方式转换到满屏文本显示方式。

格式: TEXT

3.8 高分辨率彩色作图

3.8.1 HGR

功能: 置屏幕为高分辨率图形与文本显示混合方式。显示第一页。

格式:HGR

说明:执行 HGR 后,将屏幕上图形区清为黑色,如果原来在第一页有高分辨率图象,则图象也被清除。从混合显示方式转到纯高分辨率图象显示方式及再返回来时,也是分别访问地址 -16302 和 -16301。

3.8.2 HGR2

功能:同 HGR 一样,仅使用第二页。

格式:HGR2

3.8.3 HCOLOR

功能:置高分辨率图象的颜色。

格式:HCOLOR =n

说明:在高分辨率图形方式下有 8 种颜色。n 值所对应的颜色编码如下表:

n 值	颜色
0	黑
1	绿
2	蓝
3	白 1
4	黑 2
5	红
6	黄
7	白 2

3.8.4 HPLOT

功能:在高分辨图象方式下画图。

格式:HPLOT X,Y

HPLOT TO X2,Y2

HPLOT X1, Y1 TO X2, Y2 〔{TO Xi, Yi}〕

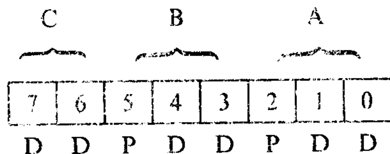
i=3, 4, 5, …

说明：格式 1 表示在(X, Y)坐标位置画一个点。格式 2 表示以上一次所画的最后一个点到(X, Y)坐标之间画一条线。格式 3 表示从(X1, Y1)到(X2, Y2)之间画一条线，再从(X2, Y2)到(X3, Y3)画一条线，依次画下去，它的范围受屏幕的限制，一条指令最多有 239 个字符的限制。X 坐标值在 0 ~ 279 之间取值，Y 坐标在 0 ~ 159 之间取值。如果在满屏图形方式下，Y 坐标值可在 0 ~ 191 之间取值，X 和 Y 以及 Xi 和 Yi 都可以由算术表达式来代替。

3.9 高分辨率图形的向量作图法

在高分辨率作图方式下还有另一种画图方法称作向量作图法。这种方法的大致原理是：将绘制的图形用一组"描绘向量"将其轨迹描绘出来，再将这组向量存放在内存的某个区域。这一组"描绘向量"就构成内存中的"一张"图形定义表。然后使用特定的命令调用这张图形定义表中的"描绘向量"，图形就可以展现在屏幕上。

存放"描绘向量"的内存区中，每个字节被分成三段。如下图所示：



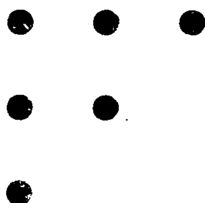
其中最低三位为 A 段；3 ~ 5 位为 B 段；最高二位为 C 段。

A、B 两段中三位数的低二位及 C 段中的二位数称做 D 位；A、B 两段中的高位称 P 位。对于 D 和 P 做如下的规定：

$$P = \begin{cases} 0 & \text{不画点} \\ 1 & \text{画点} \end{cases}$$

$$DD = \begin{cases} 00 & \uparrow \\ 01 & \rightarrow \\ 10 & \downarrow \\ 11 & \leftarrow \end{cases}$$

这表示，一个段中的 P=1 时，按此段作图时就先画一个点，然后根据 D 位的数移动一个坐标点(按高分辨作图时，屏幕上坐标点共可有 192 × 280)；若 P=0 或者无 P 位(C 段)时，则根据 D 位的数，仅移动一下坐标点。例如，要画出下图(A)中的 6 个点，可以用下图(B)中表示的 a ~ g 七个步骤来完成：


图 A

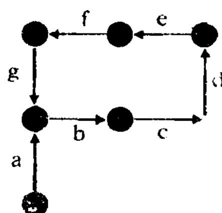


图 B

其中 a 表示画一个点，然后上移一个坐标；b 表示画一个点，然后右移一个坐标；c 与 b 相同。但 d 则表示不画点，仅上移一个坐标。因此，a ~ g 对应的描绘向量段应分别为：

a: 100

b: 101

c: 101

d: 000(或 00)

e: 111

f: 111

g: 110

但是,把各个段组成“描绘向量”时还要考虑下面的规定:

(1) 当画图时,顺序从图形定义表中逐个将“描绘向量”取出来,按其含意,由 A ~ C 的顺序作图。

(2) 对每个“描绘向量”来说,如果 C=00,则 C 段不起作用;如果 B 也为全 0,则仅 A 段起作用。

(3) 没有 A、B、C 三段全为 0 的“描绘向量”。这种字节仅用来表示图形定义表的结尾。

因此,仍以上图为例,其“描绘向量”应为下图所示,共占 4 个字节。

	C 段		B 段			A 段		
向量 1	0	0	1	0	1	1	0	0
向量 2	0	0	0	0	0	1	0	1
向量 3	0	0	1	1	1	0	0	0
向量 4	0	0	1	1	0	1	1	1

在使用时,内存中可以存放多个图形定义表。每个表中除最后一个单元为全 0 字节(表示表的结尾)外,均为“描绘向量”。这些表的总数及存放位置由另一个表——索引表表示。下图即为有 n 个图形定义表时,索引表的安排。

S	图形定义表总数 n	(0 < n < 255)
S+ 1	无用	
S+ 2	D1L	1 号图形定义表首址 D1
S+ 3	D1H	
S+ 4	D2L	2 号图形定义表首址 D2
S+ 5	D2H	
⋮	⋮	
S+ 2n	DnL	n 号图形定义表首址 Dn
S+ 2n+ 1	DnH	

图中 S 是索引表的首地址，表中主要存放了各个图形定义表中第一个向量存放单元相对于 S 的相对地址。而每个图形定义表，从这里开始存放"描绘向量"字节。存放完后，再存放一个全 0 字节。

这样，根据索引表便可逐次地找到各个图形定义表。找到每个表后，按上面说的规定逐段作图，直到遇到一个全 0 字节为止。最后便可把 n 个表所要描绘的总的图形画在屏幕上。

当要使用这种向量作图时，要先把索引表的首址(S)存放到＄E8 和＄E9 单元中(＄E8 存其低位，＄E9 存放其高位)。

BASIC 提供下述四个命令，利用它们在图形定义表的基础上可画出各种图形(包括放大或缩小的图形，旋转后的图形等等)。

3.9.1 DRAW

功能：根据某一图形定义表在屏幕上进行作图。

格式:DRAW n AT X,Y

DRAW n

说明:本命令将根据第 n 个图形定义表作图。(X,Y)为图形起点的坐标值。其中:0 < n < 255, 0 < X < 279 且 0 < Y < 191。在使用第二种格式时,其起点定义为上次使用 DRAW、HPLOT 或 XDRAW 时,所画图形的最后一个点的位置坐标。若前面没有使用这些命令,则认为它为(0,0)。

3.9.2 XDRAW

功能:清除 DRAW 语句所画的图形。

格式:XDRAW n AT X,Y

XDRAW n

说明:此命令格式及使用方法与 DRAW 一样,但作用相反,它是对图形进行删除。例如,用 DRAW 画一个图形,然后用 XDRAW 画同一图形(命令参数都一样),则将该图形清除了。或者说,第二个图形是用第一个图形的补色沿同一轨迹画一遍(补为黑色)。

3.9.3 SCALE

功能:决定用 DRAW 或 XDRAW 命令作图时的比例因子。

格式:SCALE=n

说明:n 应取值在 0 到 255 之间。当 n=1 时,即如前面所说的,一个向量段只能移一个坐标点;n=2 ~ 255 时,此移动量放大了对应的几倍;当 n=0 时,相当于 n=256,图形将放大 256 倍。

3.9.4 ROT

功能:决定由 DRAW 或 XDRAW 命令作图时,图形的旋转度。

格式: ROT =m

说明: m 值可取 0 ~ 255 的任何数，但大于 64 时，按 64 取模，m 值的含意与 SCALE 命令中的 n 值有关，表示如下:

n =
- 1
  - m = 0　右旋 0 度
  - m =16　右旋 90 度
  - m =32　右旋 180 度
  - m =48　右旋 270 度
- 2　根据 m 可识别 8 种旋转度
- ＞5　根据 m 可识别 64 种旋转度

下面，给出一段程序，运行时可以得到一个动态的运动图形。可以看出，其索引表首址是在 $ 300(=768)，共有五个图形定义表。

```
20   DATA 5,0,12,0,21,0,30,0,39,0,46,0
30   DATA 45,40,32,5,168,174,21,37,0
40   DATA 45,40,32,5,40,45,21,37,0
50   DATA 45,5,40,5,40,45,21,37,0
60   DATA 45,45,45,45,45,37,0
70   DATA 9,5,32,5,40,168,21,21,173,37,0
80   FOR I =768 TO 824
90   READ X: POKE I,X
100  NEXT I
110  POKE 232,0: POKE 233,3
120  HOME: HGR: HCOLOR =3: SCALE =1:
     ROT =0
```

```
130   FOR X=10 TO 200 STEP 4
140   FOR I=1 TO 5
150   DRAW I AT X,100
160   XDRAW I AT X,100
170   NEXT I
180   NEXT X
190   GOTO 130
```

3.10 系统实用命令与控制字符

3.10.1 LOAD

功能:将磁带上A类文件装入内存。

格式:LOAD〔文件名〕

说明:见前面增加的BASIC语句和功能一节。

3.10.2 SAVE

功能:将内存中的BASIC程序存入磁带。

格式:SAVE〔文件名〕

说明:见前面增加的BASIC语句和功能一节。

3.10.3 RUN

功能:运行一个BASIC程序。

格式:RUN

RUN 行号

说明:第一种格式是运行在内存中的BASIC程序。第二种格式是从指定行号开始运行内存中的BASIC程序。

3.10.4 NEW

功能:删除当前内存中BASIC程序及所有变量。

格式:NEW

格式: LIST

LIST〔行号 1〕〔,行号 2〕

说明: 第一种格式是列出全部程序清单。第二种格式是从行号 1 列到行号 2。

3.10.6 HOME

功能: 清屏幕,光标回到左上角。

格式: HOME

3.10.7 STOP

功能: 中止程序执行。

格式: STOP

说明: 执行 STOP 命令后,系统提示"BREAK IN 行号",系统控制权交还给用户。

3.10.8 END

功能: 使程序终止执行,并且不输出任何提示信息。

格式: END

3.10.9 CTRL-C 键

功能: 中断程序执行。

说明: 按下 CTRL 键,同时按下 C 键。

3.10.10 CTRL-S 键

功能: 暂停向上滚动的屏幕显示信息。

说明: 按下 CTRL 键,同时按下 S 键。

3.10.11 CONT

功能: 如果程序是由 STOP、END 或 CTRL-C 停止运行的,那么 CONT 命令可以使程序从下一个语句开始继续运行。

3.10.12 IN

功能:选择输入设备。

格式:IN #n

说明:n 表示输入通道号。它的取值只能是 0 ~ 7 之间的整数。当 n=6 时,启动磁盘驱动器进行读盘操作,即执行磁盘驱动程序(该程序装在第 6 号槽上);当 n=3 时,进入汉字状态;当 n=0 时,返回以键盘作为输入设备的状态;在其它情况下,则将槽口所连的输入设备以 n 作为槽号设定为后继输入的输入设备。进入汉字系统后,不能使用此语句。

3.10.13 PR

功能:选择输出设备。

格式:PR #n

说明:n 表示输出通道号。它的取值只能是 0 ~ 7 之间的整数。当 n=3 和 6 时,其作用与 IN 中介绍的相同;当 n=0 时,返回以 CRT 作为输出设备的状态;在其它情况下,则将槽口所连的输出设备以 n 作为槽号设定为后继输出的输出设备(CRT 指显示器或电视机)。进入汉字系统后,不能使用此语句。

附录A 出 错 信 息

在一个错误产生后，BASIC 返回到命令一级上，这由提示字符及一个闪烁的光标指出。变量和程序正文仍然是完整的，但是程序不能继续执行，并且所有的 GOSUB 和 FOR 循环计数器都置成 0，要在一个运行程序中避免这种中断，则可以把 ONERR GOTO 语句与错误处理例行程序结合起来使用。

当一个错误产生在一个立即执行语句中时，行号不被打印出来。出错信息的格式下：

立即执行方式的语句　　　?XX ERROR

间接执行方式的语句　　　?XX ERROR IN YY

在上面的两个例子中，"XX"表示错误名，"YY"是产生错误的语句行号。一个间接执行语句中的错误，只有该语句被执行时才被发觉。

以下是可能产生的出错码以及它们的含义：

CAN'T CONTINUE

试图继续执行一个不存在的程序，或在产生出错后，试图继续执行程序，或者从程序中删除或增加一行之后，试图继续执行程序。

DIVISION BY ZERO

除数为零出错。

ILLEGAL DIRECT

你不能将 INPUT，DEF FN，GET，DATA 语句作为立即执行命令使用。

ILLEGAL QUANTITY

传送给数学函数或串函数的参数超出范围，产生 ILLEGAL

QUANTITY 错误的情况可能是:

(a) 负数为数组下标变量(例如:LET A(−1)=0)。

(b) 用负数或零作为 LOG 的参数。

(c) 用负数作 SQR 的参数.

(d) 在 A ∧ B 中, A 为负数, 且 B 不是一个整数。

(e) 在 MID$, LEFT$, RIGHT$, WAIT, PEEK, POKE, TAB, SPC, ON… GOTO 或作图的任一函数中使用了不恰当的参数。

NEXT WITHOUT FOR

在一个 NEXT 语句中的变量名与当前的 FOR 语句中的变量名不一致,或者无变量名的 NEXT 语句不与任何 FOR 语句相对应。

OUT OF DATA

执行一个 READ 语句时,程序中的所有 DATA 语句早已被读完,程序试图要读更多的数据,但程序中所包含的数据不够。

OUT OF MEMORY

以下的任何一种情况都会产生这种信息:程序太长;变量太多;FOR 循环嵌套多于 10 层的深度;GOSUB 的嵌套深度超过 24 层;表达式太复杂;圆括号嵌套深度多于 36 层;企图置的 LOMEM: 值太高;企图置的 LOMEM: 值低于当前值;企图置的 HIMEM: 值太低.

FORMULA TOO COMPLEX

执行到多于两条 IF "XX" THEN 形式的语句。

OVERFLOW

一个计算的结果太长,以致于不能用 BASIC 的数据格式表示,如果出现下溢出,给出一个零作为结果,并且继续执行,没有任何错误信息打印出来。

REDIM'D ARRAY

对一个数组两次使用 DIM 语句时,产生此错误。这个错误经常发

生在这种情况下:如果直接使用像 A(I)=3 这样的语句,则 BASIC 自动定义其标量长度为 10,但在程序中又跟有如 DIM A(100)的语句。如果你想要找出一个数组定义语句所在的程序行,这个出错信息是很有用的。这只要在第一行上插入该数组的定义语句 DIM,运行这个程序,则 BASIC 将告诉你原来的 DIM 语句是在什么位置上。

RETURN WITHOUT GOSUB

遇到了一条 RETURN 语句,但是没有执行相应的 GOSUB 语句。

STRING TOO LONG

试图通过使用连接运算产生一个多于 255 个字符的字符串。

BAD SUBSCRIPT

试图访问一个超出数组大小范围的数组元素,如果在一个数组的访问中使用了错误维数,这种出错信息亦产生。例如:当 A 已经用 DIM A(2,2)定义了,但出现了 LET A(1,1,1)=Z 的语句。

SYNTAX ERROR

在一个表达式中,丢掉了圆括号,在一行中有不合法的字符,不正确的标点符号等等,均会产生此信息。

TYPE MISMATCH

一个赋值语句的左边是一个数值变量,而右边是一个串,或者左边是一个串变量,而右边是一个数值型数据,另一种情况是一个函数的参数类型是一个串,但给出的却是一个数值型数据,或者反过来。

UNDEF'D STATEMENT

试图使用 GOTO、GOSUB 使 THEN 转到一个行号不存在的语句上去。

UNDEF'D FUNCTION

调用了还未定义的用户定义函数。

附录B 存储空间的节省

一、空间的提示

为了使你的程序适合于在配置较小的内存的系统上运行，下面的提示是会有用的。然而，前面两条节省内存空间的方法仅当面对严重的内存紧张的情况下，才予考虑。认真的程序员经常将他的程序保留有两种版本，一种是扩充型的版本，它带有大量文件说明(即带有大量 REM)；另一种是"压缩过的"版本，仅占用最少的内存空间。

(1) 每一行上使用多条语句，使得只有少量的空间(5 个字节)与程序中的每一行相关连。这五个字节中的两个字节以 2 进制形式存放了该行的行号。这意味着，在你的行号中不论有多少数字(最小数字是 0，最大数字是 65529)，它占用同样的字节数(两个)。尽可能地在一行上放入更多的语句将减少你的程序所使用的字节数(一个独立行可包含有 239 个字符)。

注意：在一行上联合许多语句，使得编辑以及其它修改都非常困难。这不仅对其它人，对你自己今后回过来看看程序时亦是非常难读和难以理解的。

(2) 删除所有的 REM 语句，每个 REM 语句使用的字节数至少为一个字节加上通常的正文的字节数。例如：语句 130 REM THIS IS A COMMENT 占用了 24 个字节的内存。语句 140X=X+Y：REM UPDATE SUM 中的 REM 语句占用了 12 个字节的内存，包括了 REM 前面的冒号。

注意：和多行程序一样，一个程序没有详述的 REM 语句，不仅对别人，对你自己今后回过来看程序时亦是非常难读和难于理解的。

(3) 尽可能地使用整型数代替实型数(参看本附录后面的存储分配

信息)。

(4) 用变量代替常量。假定你在程序中使用常量3.14159十次,如果你在程序中插入一条语句10 P=3.14159,并在每次需要用时,以P代替3.14159,你将节省40个字节,这亦将引起速度的提高。

(5) 一个程序不需要用END来终止,因此,在程序的末端的END语句可被删除。

(6) 重复使用相同的变量。如果你有一个临时变数T,用于在一部分程序中保留一个临时结果,则在程序的后面部分中又需要临时变量时,可再使用它。如果你要求计算机的用户在程序执行期间,在两个不同的时间上,用YES或NO来回答两个不同的问题,则可使用同样的临时变量A $存储这个回答。

(7) 用GOSUB执行完成相同作用的程序语句段。

(8) 使用矩阵的零元素。例如:A(0),B(0,X)

(9) 当A $ = "CAT"被赋值为A $ = "DOG"时,老的串"CAT"不从存储器中抹去。在你的程序中定期地使用形如X=FRE(0)的语句,将使CEC-BASIC从存储的顶部开始"内部清洗"老的字串。

二、存储的分配

简单的(非数组)实型、整型或串变量,象V,V%或V $使用7个字节。实型变量用2个字节作为变量名,用5个字节作为其值(1个指数,4个尾数);整数变量用2个字节作为变量名,2个字节作为其值,其余3个字节为0;串变量用2个字节作为变量名,1个字节为串的长度,2个字节为指向内存中串的位置的指针,其余2个字节为0。请参见CEC-BASIC变量分配图。

实型数组变量最少使用12个字节:两个字节为变量名,两个字节为数组的大小,一个为维数,每一维的大小使用两个字节,每个数组元素使用五个字节。整型数组变量的每个数组元素使用两个字节。串数组变量的每个元素使用三个字节:一个字节作为长度,二个字节作为指针,

见 CEC-BASIC 变量分配图。

串变量，无论是简单的还是数组，对串中的每个字符使用内存的一个字节。串本身是按照在程序中的先后次序来定义的，从 HIMEM：开始存放。

当使用一个 DEF 语句定义了一个新的函数时，6 个字节被用来存储指向定义的指针。

保留字如 FOR，GOTO 或 NOT 以及内部函数的名，如 COS，INT 以及 STR $ 仅占据程序存储的一个字节，程序中所有其它字符，每个字符占用一个字节。

当一个程序开始执行时，在栈中空间是按下列规则动态分配的：

(1) 每个有效的 FOR … NEXT 循环用 16 个字节。

(2) 每个有效的 GOSUB（还没有执行到 RETURN）用 6 个字节。

(3) 对一个表达式中的每对圆括号用 4 个字节，以及表达式中的每个计算出的结果用 12 个字节。

附录 C　提高程序执行速度

下面的提示将改进你的 BASIC 程序的执行时间。注意：其中某些提示同样可用来减少你的程序所占用的内存空间。这意味着在许多情况下，你可以提高你的程序执行速度。同时，你还可以改进内存的使用效率。

(1) 这大概是最重要的速度提示（速度可提高十分之一）：使用变量代替常量。把一个常量转换成它的浮点（实型数）表示要比读取一个简单变量或数组变量花费的时间多。尤其是在 FOR … NEXT 循环内部或在其它反复被执行的编码中这是非常重要的。

(2) 在 BASIC 程序的执行期间，首先遇到的变量是从变量表的初始位置开始分配的变量。这意思是，例如一个语句 5A=0: B=A: C=A 在变量表中第一个位置放 A，第二个放 B，第三个放 C(假定第 5 行是程序中一个执行的语句)。在后面的程序中，当 BASIC 碰到一个引用变量 A 时，它将只搜索变量表的第一项就找到了 A，搜索第二项可找到 B，搜索第三项才找到 C 等等。

(3) 使用不带下标变量的 NEXT 语句，NEXT 比 NEXT I 稍微快一点，因为，不要检查在 NEXT 中指出的变量是否与最近的仍然有效的 FOR 语句中变量一样。

(4) 程序执行期间，当 BASIC 遇到一个新行引用时，例如："GOTO 1000"它将从起始行开始扫描整个用户程序，直到找出所要引用的行号为止(在这个例子中是 1000)，因此，经常引用的行在程序中要尽可能地放前一些。

附录 D　保留字及内码的 10 进制值

一、对照表

10 进制值	保留字	10 进制值	保留字
128	END	134	DIM
129	FOR	135	READ
130	NEXT	136	GR
131	DATA	137	TEXT
132	INPUT	138	PR #
133	DEL	139	IN #

10 进制值	保留字	10 进制值	保留字
140	CALL	165	ONERR
141	PLOT	166	RESUME
142	HLIN	167	RECALL
143	VLIN	168	STORE
144	HGR2	169	SPEED=
145	HGR	170	LET
146	HCOLOR=	171	GOTO
147	HPLOT	172	RUN
148	DRAW	173	IF
149	XDRAW	174	RESTORE
150	HTAB	175	&
151	HOME	176	GOSUB
152	ROT=	177	RETURN
153	SCALE=	178	REM
154	SHLOAD	179	STOP
155	TRACE	180	ON
156	NOTRACE	181	WAIT
157	NORMAL	182	LOAD
158	INVERSE	183	SAVE
159	FLASH	184	DEF
160	COLOR=	185	POKE
161	POP	186	PRINT
162	VTAB	187	CONT
163	HIMEM:	188	LIST
164	LOMEM:	189	CLEAR

10 进制值	保留字
190	GET
191	NEW
192	TAB(
193	TO
194	FN
195	SPC(
196	THEN
197	AT
198	NOT
199	STEP
200	+
201	−
202	*
203	/
204	∧
205	AND
206	OR
207	>
208	=
209	<
210	SGN
211	INT
212	ABS
213	USR
214	FRE
215	SCRN(
216	PDL
217	POS
218	SQR
219	RND
220	LOG
221	EXP
222	COS
223	SIN
224	TAN
225	ATN
226	PEEK
227	LEN
228	STR $
229	VAL
230	ASC
231	CHR $
232	LEFT $
233	RIGHT $
234	MID $
235	MUSIC
236	PLAY
237	LG

二、保留字

&

ABS	AND	ASC	AT	ATN
CALL	CHR $	CLEAR	COLOR =	CONT
	COS			
DATA	DEF	DEL	DIM	DRAW
END	EXP			
FLASH	FN	FOR	FRE	
GET	GOSUB	GOTO	GR	
HCOLOR =	HGR	HGR2	HIMEM:	HLIN
	HOME	HPLOT	HTAB	
IF	IN#	INPUT	INT	INVERSE
LEFT $	LEN	LET	LIST	LOAD
	LOG	LOMEM:	LG	
MID $	MUSIC			
NEW	NEXT	NORMAL	NOT	NOTRACE
ON	ONERR	OR		
PDL	PEEK	PLOT	POKE	POP
	POS	PRINT	PR#	PLAY
READ	RECALL	REM	RESTORE	RESUME
	RETURN	RIGHT $		
	RND	ROT =	RUN	
SAVE	SCALE =	SCRN(	SGN	SHLOAD
	SIN	SPC(		
	SPEED =	SQR	STEP	STOP

	STORE	STR $		
TAB(	TAN	TEXT	THEN	TO
	TRACE			
USR				
VAL	VLIN	VTAB		
WAIT				
XPLOT	XDRAW			

BASIC“能辨认出”的这些保留字，每个字仅占用程序存储的一个字节，所有其它的字符在程序存储器中各占用一个程序存储字节。见附录 D 的保留字值。

连接符(&)打算仅作为计算机内部使用，它不是一个正规的 BASIC 命令，这个符号当作为一条指令执行时，使其无条件地转到 $ 3F5 的位置上。

XPLOT 是一个保留字，不作为当前的一个 BASIC 命令。

对于某些保留字，CEC-BASIC 要根据其上下文才能识别：

COLCR，HCOLOR，SCALE，SPEED 和 ROT

仅当下一个非空格字符是赋值符号“=”时，它们才作为保留字，在 COLOR 和 HCOLOR 的情况下，保留字 OR 会妨碍它们用作任何变量名。

SCRN SPC 和 TAB

仅当下一个非空格字符在圆括号中时，它们才作为保留字。

HIMEM：如果它作为一个保留字，那么，也必须有一个冒号(:)。

ATN 仅当在 T 和 N 之间没有空格，它才作为保留字，如果在 T 和 N 之间存在一个空格，那么 AT 做为一保留字，而不是 ATN。

TO 除非前面有 A 并且在 T 和 O 之间无空格，否则把它们作为保留字，如果 T 和 O 之间存在一空格，那么把 AT 作为保留字，而不是把 TO 作为保留字。

有时圆括号可用来回避保留字:

100 FOR A=LOFT OR CAT TO 15

用 LIST 列出来为:

100 FOR A=LOF TO RC AT TO 15

但是 100 FOR A=(LOFT) OR (CAT)TO 15

用 LIST 列出来,则仍然是:

100 FOR A=(LOFT) OR (CAT) TO 15

附录 E　PEEK、POKE 和 CALL 的用法

这儿是一些你以借助于 PEEK, POKE 或 CALL 的命令来使用 BASIC 的特殊性能。注意:其中某些与 BASIC 中其它一些命令有完全相同的效果。

简单的开关动作是依赖于地址的,任何包含那个地址的命令对那个开关将起作用。例如:POKE−16304, 0 就是一个例子. 对于这个例子,你可以通过使用 POKE 来把从 0 到 255 中的任何数放到那个地址中,也可通过 PEEK 来对那个地址进行读取:X=PEEK(−16304)来获得相同的效果。

这对某些命令不适用。在这些命令中,你必须使用 POKE 来把一个特定的值放到所需的地址中,这些特定的值可以置一个页边的空白,或者用来把光标移到一个指定的地方。

一、置文本显示窗口

在下面例子中行号为 10, 20, 30 和 40 的前四个 POKE 命令是用于置 TV 屏幕上的“窗口”大小。在这个窗口上,显示正文,并且能够自动“上滚”。这些命令分别地置窗口的左边空白,行的宽度,窗口的顶部空白和底部空白。

10 POKE 32,L

置左边的空白，其大小由 L 指出的值确定，值的范围在 0 到 39 之间，这儿的 0 是指最左边的位置。

窗口的宽度不是由这条命令来改变的，这意味着右边的空白是根据你移动左边空白的同样数目来变化。为了保护你的程序及 BASIC，首先应适当压缩窗口的宽度，然后改换左边空白。

20 POKE 33,W

TV 显示的宽度（每行字符的个数）为 W 指出的值，这个值在 1 到 40 的范围内。

不能置 W 为零：POKE 33,0 会破坏 BASIC。

如果 W 小于 33，PRINT 命令的第三个制表（TAB）域可能把字符打印在窗口外。

30 POKE 34,T

根据 T 指定的值建立 TV 显示的顶部空白，其 T 的范围在 0 到 23 之间，这儿的 0 是屏幕的顶行。POKE 34, 4 将不允许在屏幕的前四行上打印文本。所置的窗口的顶部的空白（T）不能低于底部的空白（下面的 B）。

40 POKE 35,B

根据 B 指定的值建立屏幕底部空白；其 B 的范围是 0 到 24 之间。这儿的 24 是屏幕底部的一行，窗口底部的空白（B）不能高于顶部空白（即上面的 T）。

二、有关文本显示、文本显示窗口和键盘的一些命令

45 CALL-936

清除文本显示窗口里面的所有字符，且移动光标到窗口的左上

50 CALL-958

清除在文本窗口里从当前光标位置到底部空白处的所有字符，在当

前行上光标上面的字符以及光标左边的字符将不受影响。

60 CALL-868

清除当前行从光标位置到右边空白。

70 CALL -922

发出一个换行。

80 CALL-912

上滚正文一行，即在所定义的窗口内的每行正文都向上移动一行位置，老的顶端的行丢失。老的第二行成为第一行。底端的一行现在是空白。在所定义的窗口外的字符不受影响。

90 X=PEEK(-16384)

读键盘，如果 X >127 那么表示已按了一个键，X 的内容是所按键的 ASCII 码的值，且第 7 位为 1。这在长程序中是有用的。在这个程序中，计算机检查用户是否想要用新数据去中断一下程序，而不是停止程序的执行。

100 POKE -16368,0

复位键盘选通，以便下一个字符可以读入，这在读入键盘后应立即去做。

三、与光标有关的命令

110 CH=PEEK(36)

读回当前光标的水平位置，并且把这个值赋给变量 CH。光标将在 0 到 39 的范围内，且是与文本显示窗口的左边空白有关的光标的位置。这个左边空白是由 POKE 32,L 置的，因此，如果由 POKE 32,5 置了左边空白，那么，窗口中最左边的字符是在从屏幕的左端起的第六个打印位置上，且如果 PEEK(36)送回的值是 5，那么，光标是在从屏幕的左边缘起第十一个打印位置上，即在从文本显示窗口的左起第六个打印位置上(最初听起来有些混乱，因为最左边的位置是 0，而不是 1)。这与 POS(X)函数是相同的(参看下一个例子)。

120 POKE 36,CH

移动光标到从文本显示窗口的左起第 CH+1个打印位置上(例:POKE 36,0将使得下一个字符从窗口的左边打印)。如果窗口左边空白在用 PRINT 打印之前必须改变,CH 必须小于或等于由 POKE 22,W 置的窗口的宽度,且必须大于或等于 0。象 HTAB 一样,这命令能够把光标移出正文窗口的右边空白,但仅够打印一个字符。

130 CV=PEEK(37)

读出当前光标的垂直位置,且置 CV 等于该值,CV 是光标的绝对垂直位置,它与正文屏幕的顶部或底部的空白无关。因此,CV=0 是屏幕的顶端行,CV=23 是底行。

140 POKE 37,CV

移动光标到由 CV 指定的绝对垂直位置上。0 是最上面的一行,23 是底行。

四、有关作图的命令

为了显示文本和制图,中华学习机的内存划分出四个区域:文本显示的第 1 页和第 2 页以及高分辨率图形显示的第 1 页和第 2 页。

(1) 文本显示的第 1 页是所有的文本和低分辨制图的通用的内存区域,这通过 TEXT 和 GR 命令来使用。

(2) 在内存中文本显示的第 2 页刚好位于文本显示的第 1 页的上面。对用户来讲,这页是不容易存取到的。同正文第 1 页一样,存储在正文第 2 页上的信息可以解释为正文,也可以解释为低分辨率图形,或者两者皆可。

(3) 高分辨率图形的第一页驻留在中华学习机的内存的 8K 到 16K 上,这个区域是通过 HGR 命令来使用的。

(4) 高分辨率图形的第 2 页驻留在中华学习机内存的 16K 到

24K 上。这个区域是通过 HGR2 命令使用的。

你可以用 BASIC 的正文和制图命令或者用这四种不同的开关来使用不同的制图和正文方式。如同这儿讨论的许多开关一样。PEEK 或 POKE 可对一个地址置开关为一种状态，以及 PEEK 和 POKE 可对第 2 个地址置开关为另一个状态，简单地说，这四种开关的选择区间为：

(1) 文本显示 (POKE−16303,0)

高分辨率或低分辨率图形显示 (POKE−16304,0)

(2) 文本或高分辨率显示的第一页 (POKE−16300,0)

文本或高分辨率显示的第二页 (POKE−16299,0)

(3) 文本显示的第一页或制图的第二页 (POKE−16293,0)

制图的高分辨率的第一页或第二页 (POKE−16297,0)

(4) 满屏幕高分辨率或低分辨率制图 (POKE−16302,0)

高分辨率或低分辨率制图加文本显示混合方式(POKE−16301,0)

150 POKE−16304,0

从文本显示方式转变到彩色制图方式中，而不清除制图屏幕为黑色。根据另外三个开关的置位，转变成制图方式，可以是低分辨或高分辨率(从第 1 页到第 2 页)，也可以是制图加文本显示的混合方式或满屏幕制图方式。

BASIC 中类似的命令：GR 命令用来转变成第一页的低分辨率制图加正文的混合屏幕，并且清除制图屏幕为黑色；HGR 命令用来转变成为第一页高分辨率制图加文本显示的混合屏幕，并且清图形屏幕为黑色；HGR2 命令用来转变成第二页高分辨率满屏图形且清除整个屏幕为黑色。

160 POKE−16303,0

从任何彩色制图显示方式转变成所有文本显示方式，而不重置"上滚"窗口，根据第一页 / 第二页开关的置位，可将文本显示页从第一页

转换成第二页，或从第二页转换成第一页。

TEXT 命令将屏幕置成全文本显示方式，它除了选择文本显示的第一页以外，还重新将窗口复位成最大，并将光标移到显示屏幕的左下角上。

170 POKE -16302,0

转换制图加文本显示的混合屏幕为满屏幕制图显示方式。

根据其它开关的置位，可以作为正文或是一个 40 乘 48 的栅格上的低分辨率制图，或一个 278 乘 192 栅格的高分辨率制图出现。

180 POKE -16301,0

从满屏图形转换成图形加文本显示的混合屏幕方式，在屏幕的底部有 4 行文本显示区，每行 40 个字符。

根据其它开关的置位，屏幕的上部分可以显示文本，也可以在一个 40 乘 48 的栅格上进行低分辨率制图或在一个 278 乘 160 栅格上进行高分辨率制图。屏幕的两部分显示将来自同样的页数(1 或 2)。

184 POKE -16300,0

从第二页转换到第一页，不清除屏幕或移动光标。当你从 BASIC 进入到整型 BASIC 时这是必须的，否则，你看到的将仍然是内存的第二页的内容。

根据其它开关的置位，这可以使得显示从高分辨率制图的第二页改变到高分辨率制图的第一页，从低分辨率制图的第二页转变为低分辨率制图的第一页，或者从文本显示的第二页转变成文本显示的第一页。

186 POKE 16299,0

从第一页转换到第二页，不清除屏幕或移动光标。

根据其它开关的置位，这可以使得显示从高分辨率制图的第一页改变成高分辨率制图的第二页；从低分辨率制图的第一页改变成低分辨率制图的第 2 页，或者从正文第一页改变成正文第二页。

(HCOLOR)（BASIC 记住最后使用的屏幕，而不考虑开关的置位）。在这前面必须有一个 HPLOT 语句。

五、与游戏控制器及扬声器有关的命令

220 X=PEEK(-16336)

触发(TOGGLE)扬声器一次，扬声器发出一个"卡嗒"声。

225 X=PEEK(-16352)

触发盒式录音带输入一次，在盒式录音机上产生一个"卡嗒"声。

230 X=PEEK(-16287)

读#0 游戏控制器上的按钮开关，如 X >127，那么按钮已被按了。

240 X=PEEK(16286)

同上面的一样，只是读#1 游戏控制器的按钮开关。

250 X=PEEK(-16285)

读#2 游戏控制器上的按钮。

六、与出错有关的命令

340 X=PEEK(218)+ PEEK(219) * 256

如果一条 ONERR GOTO 语句已执行过，这条语句置 X 等于产生错误的那条语句的行号。

350 IF PEEK(216) >127 THEN GOTO 2000

如果内存 222 单元(ERRFLG)的第 7 位已经置为真，那么，已经遇到一条 ONERR GOTO 语句。

360 POKE 216,0

清除 ERRFLG，以便发出标准出错信息。

370 Y=PEEK(222)

置变量 Y 为描述出错类型的编码，这个错误已使得一条 ONERR GOTO 转移出现。错误类型描述见 ONERR GOTO 语句。

七、简单变量

指针	实型	整型	串指针
$69～$6A	名(pos)第1个字节	名(neg)第1个字节	名(neg)第1个字节
	(pos)第2个字节	(neg)第2个字节	(pos)第2个字节
	指数 1个字节	高位字节	长度 1个字节
	尾数 m.s.byte	低位字节	地址 低位字节
	尾数	0	地址 高位字节
	尾数	0	0
	尾数 l.s.byte	0	0

串进入的顺序是从 HIMEM: 开始向下存储的，串表指向内存中每个串的第一个字符和最后一个字符。

当改变串时，重写新的指向地址，当可用的内存已用完时，内部清理删除所有的废弃的串（内部清理是通过一个 FRE(X)促成的）。

所有数组按最右边指针上升最慢的次序来存放。例如：数组 A%(1,1)中的数，其中 A%(0,0)=0，A%(1,0)=1，A%(0,1)=2，A%(1,1)=3 将在内存中以正确的顺序找出。

八、数组变量

	实型	整型	串指针
$6B～$6C	名(pos)第1个字节	名(neg)第1个字节	名(neg)第1个字节
	(pos)第2字个节	(neg)第2个字节	(pos)第2个字节
	形成指针指向下一个变量，加上这个变量名的地址	形成指针指向下一个变量，加上这个变量名的地址	形成指针指向下一个变量，加上这个变量名的地址
	低位字节	低位字节	低位字节
	高位字节	高位字节	高位字节

续表:

	维数的个数1个字节	维数的个数1个字节	维数的个数1个字节
	第 N 维的大小	第 N 维的大小	第 N 维的大小
	高位字节	高位字节	高位字节
	低位字节	低位字节	低位字节
	第 1 维的大小	第 1 维的大小	第 1 维的大小
	高位字节	高位字节	高位字节
	低位字节	低位字节	低位字节
	REAL(0,0 … 0)	INTEGER %	STRING $
	指数 1个字节	(0,0 … 0)	(0,0 … 0)
	尾数 m.s.byte		长度 1个字节
	尾数	高位字节	地址 低位字节
	尾数		
	尾数 1.s.byte	低位字节	地址 高位字节
	REAL(N,N … N)	INTEGER %	STRING $
	指数 1个字节	(N,N … N)	(N,N … N)
	尾数 m.s.byte		长度 1个字节
	尾数	高位字节	地址 低位字节
	尾数		
$ 6D ~ $ 6E	尾数 1.s.byte	低位字节	地址 高位字节

附录F　零页的使用

单元位置(16进制)	说　　明
\$ 0—\$ 5	继续CEC-BASIC的转移指令，(CEC-BASIC的RESET 0G RETURN等于整型BASIC的RESET CTRL-C RETURN)。
\$ A—\$ C	USR函数的转移指令地址见USR函数描述。
\$ D—\$ 17	通常用于CEC-BASIC的计数器／标志。
\$ 20—\$ 4F	紫金Ⅱ A系统监控的保留单元。
\$ 50—\$ 61	通常用作CEC-BASIC的指针。
\$ 62—\$ 66	上次乘法／除法的结果。
\$ 67—\$ 68	指向程序开始的指针，对ROM版本来讲通常置成\$ 0801，或对RAM(盒式录音带)版本来讲通常置成\$ 3001。
\$ 69—\$ 6A	指向简单变量区域开始位置的指针，亦指向程序的结束，并加上1或加上2，除非用LOMEM:语句来修改。
\$ 6B—\$ 6C	指向数组区域开始位置的指针。
\$ 6D—\$ 6E	指向使用中的数值存储区域的末端的指针。
\$ 6F—\$ 70	指向串存储区域开始的指针，串可从这儿开始一直存储到内存的末端。
\$ 71—\$ 72	通用指针。
\$ 73—\$ 74	CEC-BASIC可用的内存的最高单元加1。在最初进入CEC-BASIC时，置成最高的可用的

单元位置(16 进制)	说　　明
	RAM 内存地址。
$ 75—$ 76	当前正在执行的程序行的行号。
$ 77—$ 78	“旧行号”通过 CTRL-C, STOP 或 END 语句来建立的，给出程序执行时被中断的行的行号。
$ 79—$ 7A	“旧的正文指针”指出下一次将要执行的语句在存储器中的地址。
$ 7B—$ 7C	当前 DATA 语句的行号，该行中的数据正由 READ 语句来读.
$ 7D—$ 7E	指出内存中的绝对单元地址，READ 语句正在从这个单元地址中读 DATA 中的数据。
$ 7F—$ 80	指向当前输入的源程序的指针，在 INPUT 语句执行期间置为 $ 201. 在一条 READ 语句执行期间被置成在程序中 DATA 是从中读入的单元.
$ 81—$ 82	保存上次使用的变量名.
$ 83—$ 84	指向上次使用的变量的值的指针。
$ 85—$ 9C	通用。
$ 9D—$ A3	主要的浮点累加器。
$ A4	通常用于浮点的数学例行程序.
$ A5—$ AB	次要的浮点累加器.
$ AC—$ AE	通常用于标志 / 指针。
$ AF—$ B0	指向程序的末尾(不能用 LOMEM: 改变).
$ B1—$ C8	CHRGET 例行程序. CEC-BASIC 每次希望得到另一个字符时，在这儿调用它。
$ B8—$ B9	指向通过 CHRGET 例行程序得到的上一个字符的指针。
$ C9—$ CD	随机数。

单元位置(16进制)	说　　明
$ D0—$ D58	高分辨率制图划线(Scratch)指针。
$ D8—$ DF	ONERR 指针 / 划线。
$ E0—$ E2	高分辨率制图的 X 和 Y 坐标。
$ E4	高分辨率制图的颜色字节。
$ E5—$ E7	通常用于高分辨率制图。
$ E8—$ E9	指向图形表开头的指针。
$ EA	高分辨率制图的碰撞计数器。
$ F0—$ F3	通用标志。
$ F4—$ F	ONERR 指针。

附录 G　CEC-BASIC 命令参考卡

1. 简单变量

类型	名	范围
实型	AB	+/−9.99999999 E+ 37
整型	AB %	+/−32767
串	AB $	0到255个字符

这里 A 是字母,B 可以是字母也可以是数字。名可以由二个以上的字符组成,但仅有最前面的二个字符是有意义的:AB % 和 AB3QS % 是同一整数变量。

2. 数组变量

类型	典型的元素名
实型	AB　(3,12,7)
整型	AB %　(3,12,7)
串	AB $　(3,12,7)

数组的大小是受有效内存所限制的。

3. 代数运算符

= 赋值给变量(LET 是选用项)

– 负号

∧ 次幂

* 乘

/ 除

+ 加

- 减

4. 关系和逻辑运算

= 等于

< > 不等

< 小于

<= 小于或等于

> 大于

>= 大于或等于

NOT 逻辑“非”

AND 逻辑“与”

OR 逻辑“或”

关系和逻辑表达式为真时，其值为 1，为假时，其值为 0，逻辑运算符也能够用于串的比较。

5. 系统和实用命令

LOAD 从磁带上装入程序。

SAVE 把程序存到磁带上。

NEW 删除当前程序。

RUN 从最小行号处开始运行程序。

RUN477 从 477 行开始运行程序。

STOP	停止执行并报告停止那一行。
END	停止执行，但不显示信息。
CTRL-C	用于立即执行方式来停止程序或列表。
RESET	无条件地转向监控程序。用 CTRL-C 或 0G 来返回到 CEC-BASIC。
CONT	继续执行由 STOP, END 或 CTRL-C 所终止的程序。
TRACE	提供调试手段，列出它每次新执行到的每一行的行号。
NOTRACE	关闭 TRACE。
PEEK (X)	送回存储在单元 X 的内容。
POKE X, 13	将内存单元 X 的内容修改成 13。
WAIT X, Y, Z	等待，直到 X 单元的内容与 Z 进行 XOR 运算，并且再与 Y 进行 AND 运算。所得到的值为非零时为止。
CALL X	转向以内存单元 X 开始的机器语言子程序。
USR (X)	把值 X 送给机器语言子程序。
HIMEM:	置 CEC-BASIC 程序可使用的最高的内存有效地址。
LOMEM:	置 CEC-BASIC 程序可使用的最低的内存有效地址。

6. 编辑命令和与格式有关的命令

LIST	列出整个程序。
LIST X-Y	列出从 X 行到 Y 行的程序段。
DEL X, Y	删除从 X 行到 Y 行的程序段。
REM XYZ	用来书写程序的注解，程序将这部分忽略。
VTAB Y	把光标移到 Y 行(1 到 24)。

HTAB X	把光标移到 X 位置上(1 到 40)。
TAB (X)	仅用于 PRINT 语句,它把光标移到 X 位置上(1 到 40)。
POS(0)	送回光标当前所在的水平位置的值(0 到 39)。
SPC (X)	仅用于 PRINT 语句,在最后一次打印的项和要打印的下一项之间放上 X 个空格。
HOME	清除屏幕并把光标放在顶部。
CLEAR	所有的变量重置为零。
FRE(0)	送回用户还可使用的内存总数。
FLASH	置计算机的输出为闪烁方式。
INVERSE	置计算机的输出为白底黑字方式。
NORMAL	关闭闪烁或白底黑字方式。
SPEED=X	置字符输出速率为 X(从 0 到 255)。
ESC A	把光标向右移一格。
SEC B	把光标向左移一格。
SEC C	把光标向下移一格。
SEC D	把光标向上移一格。
右箭头(→)	把光标下的字符输入内存,并把光标向右移一格。
左箭头(←)	删除当前行中刚刚打入的字符,并把光标向左移一格。
CTRL-X	删除当前打入的行。

7. 数组和串

DIM A(X,Y,Z)	置 A 的最大下标,保留 X+1 * Y+1 * Z+1 个实型元素的存储空间。由 A(0,0,0)元素开始。

DIM A $ (X, Y)	置 A$ 的最大下标，它可以包含 X+ 1 * Y+ 1 * Z+ 1 个串元素，每个元素最多可达 255 个字符。
LEN(A $)	送回 A $ 中所含的字符个数。
STR $ (X)	将 X 的数值转换成对应的字符串送回。
VAL(A $)	以 A $ 中的第一个非数字字符为界，将其前面的字符转换成数值，送回。
CHR $ (X)	送回代码为 X 的 ASCII 字符。
ASC(A $)	送回 A $ 中的第一个字符的 ASCII 代码。
LEFT $ (A $, X)	送回 A $ 中的最左边的 X 个字符。
RIGHT $ (A $, X)	送回 A $ 中的最右边的 X 个字符。
MID $ (A $, X, Y)	送回 A $ 中的从字符 X 开始的 Y 个字符。
+	用于连接串的运算符。
STORE A	把数值数组 A 保存到磁带上，不可直接用来存储串数组。
RECALL B	把数组从磁带上装回，B 数组必须已经被正确地用 DIM 定义过。

8. 输入／输出命令

(同时参考 LOAD 和 SAVE，STORE 和 RECALL)

INPUT A $	在屏幕上显示一个问号(?)等待用户打入一组字符串值给 A $。
INPUT"XYZ"; A	在屏幕上打印出 XYZ，等待用户打入一个实数值给 A。
GET A $	等待用户打入单个字符值给 A $，它无需按 RETURN 键。

DATA X, "Y", Z	建立 READ 语句能使用的数据元素表.
READ A $	将下一个 DATA 的元素赋给 A $。
RESTORE	重新从第一个 DATA 元素开始，用 READ 来读。
PRINT"X="; X	在屏幕上打印串 X=和变量 X 的值,分号表示紧接着上一项段打印,逗号则表示各打印项之间留有三个制表域(tab field)的长度。 符号?也表示 PRINT 语句。
IN# 6	从#6 插口上的外部设备中进行预输入,而不是从键盘(IN #0) 上进行。
PR * 6	置输出为#6 插口上的外部设备,而不是 TV 屏幕(PR #0)
LET X=Y	把 Y 的值赋给变量 X,LET 是选用项。
DEF FN A(X) =X+ 23 / X	定义一个函数 FNA,在后面使用时,FNA 的自变量由定义中的表达式 X 来代替。FNA(4)将送回值 9.75。

9. 有关流程控制的命令

GOTO 347	转向 347 行。
IF X=3 THEN STOP	如果断言 X=3 为真(非零),则继续执行 THEN 后的语句。如果断言为假(零)则执行下一行号的语句。
FOR X=1 TO 20 STEP 4 … NEXT X	执行 FOR 语句和相应的 NEXT 语句之间的所有语句,首先 X=1,然后 X=5,X=9 等等。直到 X >20,才继续执行 NEXT 后的

	语句。如果未指明 STEP 的大小，则表明 STEP 大小为 1。
NEXT X	定义 FOR … NEXT 循环的底，X 是选用项。
GOSUB 33	转向行号为 33 开始的子程序。
RETURN	标明子程序的结束，返回到最后的 GOSUB 的下一条语句处执行。
POP	从 RETURN 地址栈中移出一个地址。
ON X GOTO 397, 12, 458	转向表中所列出的第 X 个行号所指出的语句。如果 X=2，则转向 12 行。
ON X GOSUB 397, 12, 458	转向表中第 X 个行号的子程序。
ONERR GOTO 4500	在后继语句执行中，如产生错误使得程序转到错误处理程序(在 4500 行)上执行，而不是显示信息，也不停止程序的运行。
RESUME	在错误处理程序中，使得返回发生错误的语句处执行。

10．作图和游戏控制

低分辨率图形

GR	置成低分辨率作图方式，把顶端的 40 × 40 区域清成黑色，底部留有 4 行正文显示。
COLOR=X	为画下一个点置颜色(0 到 15)
PLOT X,Y	把有色的点显示在横坐标为 X，纵坐标为 Y 的位置上，X 和 Y 在 0 到 39 范围内，(0,0)是屏幕的左上角。
HLIN X1, X2AT Y	从点(X1,Y)到点(X2,Y)画一条水平线。

VLIN Y1,Y2AT X	从点(X,Y1)到点(X,Y2)画一条垂直线。
SCRN (X,Y)	送回在屏幕上(X,Y)这一点的颜色数。

高分辨图形

HGR	置高分辨率作图方式(第 1 页);把顶端的 280 × 160 区域清成黑色,底部留出 4 行的正文显示。
HGR 2	置第 2 页的高分辨率作图方式,按整个 280 × 192 区域的屏幕清成黑色。
HCOLOR=X	为画下一个点置颜色(0 到 7)。
HPLOT X,Y	把有色点置在横坐标为 X、纵坐标为 Y 的位置上,X 的变化范围从 0 到 279,Y 则从 0 到 159(对 HGR),或到 191(对 HGR2)。(0,0)是左上角。
HPLOT X1,Y1 TO X2,Y2	从点 X1,Y1 到点 X2,Y2 画一条线,命令还可以扩充,加上 TO Xn,Yn 等等点。
SHLOAD	从磁带上装入一个图形表。
DRAW 3 AT X Y	按照先前所装入的图形表中所定义的#3 形状来绘图,起始位置为 X,Y,所绘的颜色由 HCOLOR 来确定。
XDRAW 3 AT X Y	按照图形表中所确定的#3 形状来绘制,所画的每一点的颜色数是屏幕上那一点原有的颜色数的补数。
ROT=X	为执行 DRAW 或 XDRAW,可将形状进行旋转,ROT=0 表示垂直,ROT=16 表示顺时针方向右旋 90 度,ROT=32 表示顺时针方向右旋 180 度等等。

SCALE=X	置 DRAW 或 XDRAW 图形的比例因子(1 到 255)游戏控制。
PDL(X)	送回游戏控制器 X(0 到 3)所置的值,它可以是在 0 到 255 之间进行变化。
PEEK (X−16287)	如果其值>127,表示游戏控制器 X(0 到 2)上的按钮已被按下。
PEEK (−16336)	使 APPLE 的喇叭发出卡哒声。

11 一些常用数学函数

SIN(X)	送回 X 弧度的正弦值。
COS(X)	送回弧度 X 的余弦值。
TAN(X)	送回弧度 X 的正切值。
ATN(X)	送回 X 的反正切值,以弧度为单位。
INT(X)	送回小于等于 X 的最大整数。
RND(1)	每次使用时,总送回一个 0 到 0.999999999 的随机数。
RND(0)	再次送回上一次的随机数。
RND(−3)	送回的值为 4.48217179E−08,对于每一个不同的负自变量,总是送回相应于不同变量的一个固定值(换言之,一个函数产生的随机数是一定值,与调用次数无关)。使用了这一随机数后,带有正自变量的 RND 的函数,也产生一固定序列。
SGN(X)	如果 X <0,送回值为 −1,如果 X=0,送回值为 0,如果 X >0,送回值为 1。
ABS(X)	送回 X 的绝对值。
EXP(X)	送回 E 的 X 次方的值。e=2.718289。
LOG(X)	送回 X 的自然对数的值

SQR(X)	送回 X 的正平方根。
PLAY	装入并执行磁带游戏。
MUSIC X,Y	音乐语句,X 为频率,Y 为时间。
LG	装入并执行固化的 LOGO 语言。

第二部分
监控系统

第一章 概述

监控程序是用来管理中华学习机系统的重要程序。它已固化在 ROM 中。它通过监控命令提供对系统的基本操作,如检查、修改、传送、比较内存的数据,检查、修改 CPU 的内部寄存器,管理键盘输入、字符显示、盒式录音机存取,运行程序,跟踪程序执行的轨迹,反汇编等等。它所提供的管理系统的子程序可以为其它程序所调用,使其成为其它程序操作机器的工具。

为了进入监控状态,只需在 BASIC 状态下键入下列命令行:

CALL-151

退出监控可使用下面所介绍的 CTRL-C 或 CTRL-B 命令。

第二章　监控命令介绍

2.1　对监控命令格式的说明

监控命令的命令行中一般包含三种内容：地址、数据和命令字。其中地址和数据分别用4位和2位16进制数表示。如果监控命令中的16进制数的地址少于4位，监控程序就在它的前面加0；若多于4位，则只取后4位的16进制数。监控程序能识别22个不同的命令字符，其中包括标点符号、大写字母及控制字符。任何一个监控命令均需在键盘上输入命令行之后，再按RETURN键才能执行。监控命令执行时，显示的内容均采用16进制数表示。

监控命令行的长度受到系统键盘缓冲区大小的限制，它不得超过254个字符，否则，监控系统将从本行跳出，并忽略该行所敲入的数据及命令。

2.2　监控命令

2.2.1 显示存储器的内容

格式1：〈地址〉

功能：显示该地址单元的内容。

格式2：〈地址1〉.〈地址2〉

功能：显示从地址1到地址2之间的所有存储单元的内

容。

格式3: 〈地址2〉

功能：显示从现行的存储器位置至地址2之间的内容。

格式4: (只按一个RETURN键)

功能：显示下面8个存储单元的内容。

2.2.2 改变存储器的内容

格式1: 〈地址〉:〈数据〉␣〈数据〉␣〈数据〉……

功能：从地址单元开始顺次将所列数据送入存储器(符号"␣"表示空格,下同)。

格式2: :〈数据〉␣〈数据〉␣〈数据〉……

功能：从上次用来存放数据的地址之后开始,将数据顺次送入存储单元。

2.2.3 移动存储器中的内容

格式:〈地址1〉<〈地址2〉.〈地址3〉M

功能：将地址2至地址3区域内的数据复制到从地址1开始的存储单元中去。

2.2.4 核实存储器的内容

格式: 〈地址1〉<〈地址2〉.〈地址3〉V

功能：核实从地址2到地址3之间的数据块与从地址1开始的长度相等的数据块是否完全相同。若有不同,就显示出来。

2.2.5 磁带输入/输出

格式1: 〈地址1〉.〈地址2〉R

功能：将磁带内的数据读入所规定的存储器地址区域内。磁带数据的长度必须与存储器区域的长度(即〈地址2〉-〈地址1〉+1)相等。

格式2: 〈地址1〉.〈地址2〉W

功能:将规定的存储器区域(即〈地址 1〉至〈地址 2〉)内的数据写到磁带上。

2.2.6 置屏幕显示方式

格式 1: I

功能:置屏幕显示方式为反向方式(白底黑字)。

格式 2: N

功能:置屏幕显示方式为正常方式(黑底白字)。

2.2.7 反汇编命令

格式 1: 〈地址〉L

功能:将从指定地址开始的 20 条指令翻译成 6502 汇编记忆符,并在屏幕上显示出来。

格式 2: L

功能:将现行地址开始的 20 条指令翻译成 6502 汇编记忆符,并显示出来。

2.2.8 执行机器指令

格式 1: 〈地址〉G

功能:从指定地址开始启动机器语言程序。程序在执行完一条 BRK 指令后中断。

格式 2: CTRL-Y

功能:从 $ 3F8 开始执行用户所确定的机器语言程序。通常在 $ 3F8 处放置一条 JMP 指令,转向用户程序的入口。

2.2.9 显示及修改 CPU 寄存器

格式: CTRL-E

功能:显示累加器 A,变址寄存器 X、Y,状态寄存器 P,以及堆栈指示器 S 的内容。如要修改上述寄存器的内容,可在键入“:”之后,用空格为界,键入 1 至 5 个 16 进制数据,它们

按上述寄存器的顺序，分别取代原寄存器的内容。

2.2.10 选择输入 / 输出设备

格式 1: 〈槽号〉CTRL–P

功能：将输出控制转给槽号所指定的连接槽上的接口卡，此时槽号只能取值为 2、4、。若槽号为"0"，则返回以 CRT 为输出设备的状态。若槽号为"3"，则进入中文状态。若槽号为"6"，则转入软盘驱动程序执行。槽号的取值应小于 8。

说明：本命令只有在 $ C006 软开关置位的情况下，方能有效地使用。若槽号为 3 时，则还需 $ C00B 软开关置位。在系统 RESET 后，这两个软开关均已自动置位。本命令不能在中文状态下使用。

格式 2: 〈槽号〉CTRL–K

功能：当槽号之值为 2、4、时，将输入控制转给槽号所指定的连接槽口上的连接卡。当槽号为"0"时，则返回以键盘为输入设备的状态。在其它情况下，其功能与格式 1 相同。槽号的取值也应小于 8。

说明：同格式 1.

2.2.11 16 进制加减法运算

格式 1: 〈数据 1〉+〈数据 2〉

功能：实现以 256 为模的两位 16 进制数的加法。

格式 2: 〈数据 1〉–〈数据 2〉.

功能：实现以 256 为模的两位 16 进制数的减法。

2.2.12 退出监控

格式 1: CTRL–B

功能：使系统离开监控程序而进入 BASIC 状态，以前使用的 BASIC 程序及变量全部消失。

格式2：CTRL-C

功能：与CTRL-B相同，只是进入BASIC后，仍能保存原来的BASIC程序及变量。

2.2.13 单步执行

格式：〈地址〉S

功能：执行指定地址处的一条机器指令，并将该指令的反汇编记忆符显示出来。执行指令完后各寄存器的值也同时在屏幕上给出。如若想继续单步执行下一条机器指令，可再敲入一个S(以RETURN结束)。此过程可以一直进行下去，直到不再需要执行程序时为止。

2.2.14 跟踪执行

格式：〈地址〉T

功能：从指定地址处开始执行机器指令，每条指令执行完后，其指令的反汇编记忆符和各寄存器的值被显示出来。程序在执行完一条BRK指令后中断。

2.2.15 多重命令

监控程序允许在同一个命令行内写入多个以空格分开的监控命令，只要总字符数小于254个就可以了。但若在多重命令中使用了修改存储器内容的命令，则在该命令之后，应写一个单一字母的命令(通常用N命令)，用于把随后的命令隔开。在多重命令中，单一字母命令之间不需用空格分隔。

第 三 章
监控程序的结构分析

3.1 复位处理

中华学习机在开启主机电源或在键盘上按 CTRL-RESET 键时，都将产生一个 RESET 中断，中断处理过程如图 2.3.1。

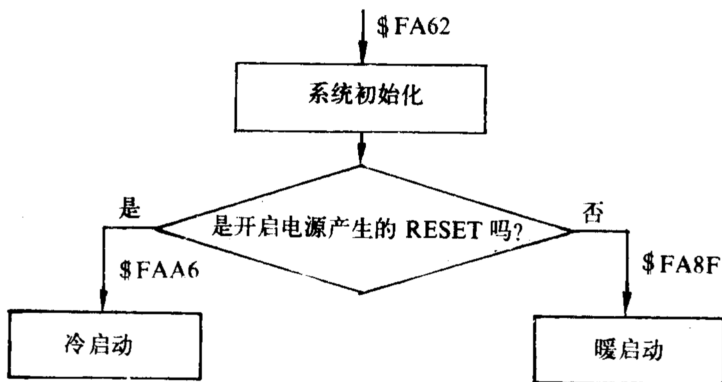


图 2.3.1 RESET 中断处理程序

3.1.1 系统初始化

图 2.3.2 表示了系统初始化的过程。

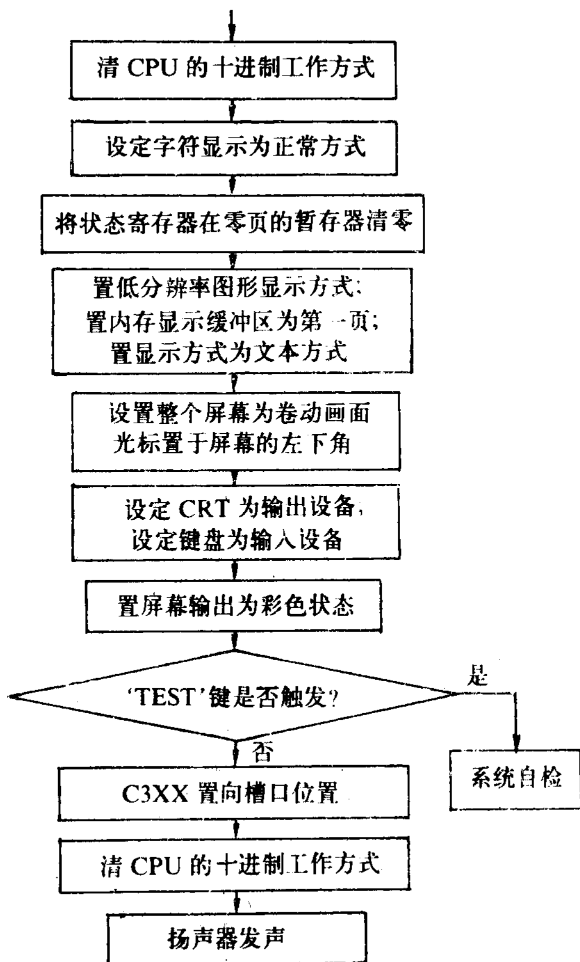

清 CPU 的十进制工作方式
设定字符显示为正常方式
将状态寄存器在零页的暂存器清零
置低分辨率图形显示方式;
置内存显示缓冲区为第一页;
置显示方式为文本方式
设置整个屏幕为卷动画面
光标置于屏幕的左下角
设定 CRT 为输出设备;
设定键盘为输入设备
置屏幕输出为彩色状态
'TEST'键是否触发?
是
系统自检
否
C3XX 置向槽口位置
清 CPU 的十进制工作方式
扬声器发声

图 2.3.2　系统初始化

3.1.2 暖启动

系统暖启动过程如图 2.3.3 所示。

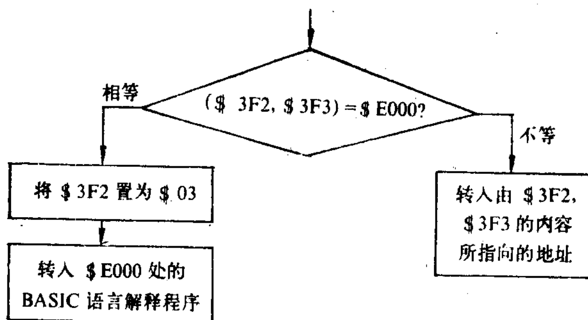


图 2.3.3　暖启动

3.1.3 冷启动

系统冷启动过程如图 2.3.4 所示。

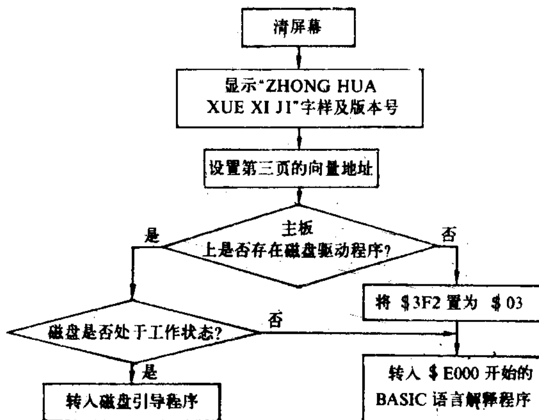


图 2.3.4　冷启动

3.2 监控命令处理器

这是监控程序的一个重要部分，进入点在 $ FF69(即65385或-151)。在BASIC状态下，若执行CALL-151，则实际上进入了监控命令处理器。监控命令处理器的框图如图2.3.5所示。

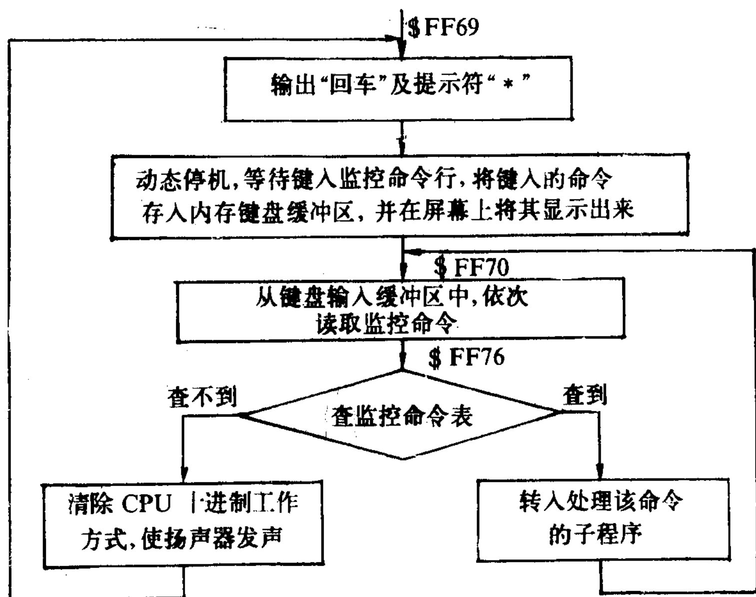


图2.3.5 监控命令处理器

3.2.1 监控命令的键入、保存与显示

图2.3.6说明了监控命令的键入、保存与显示的流程。

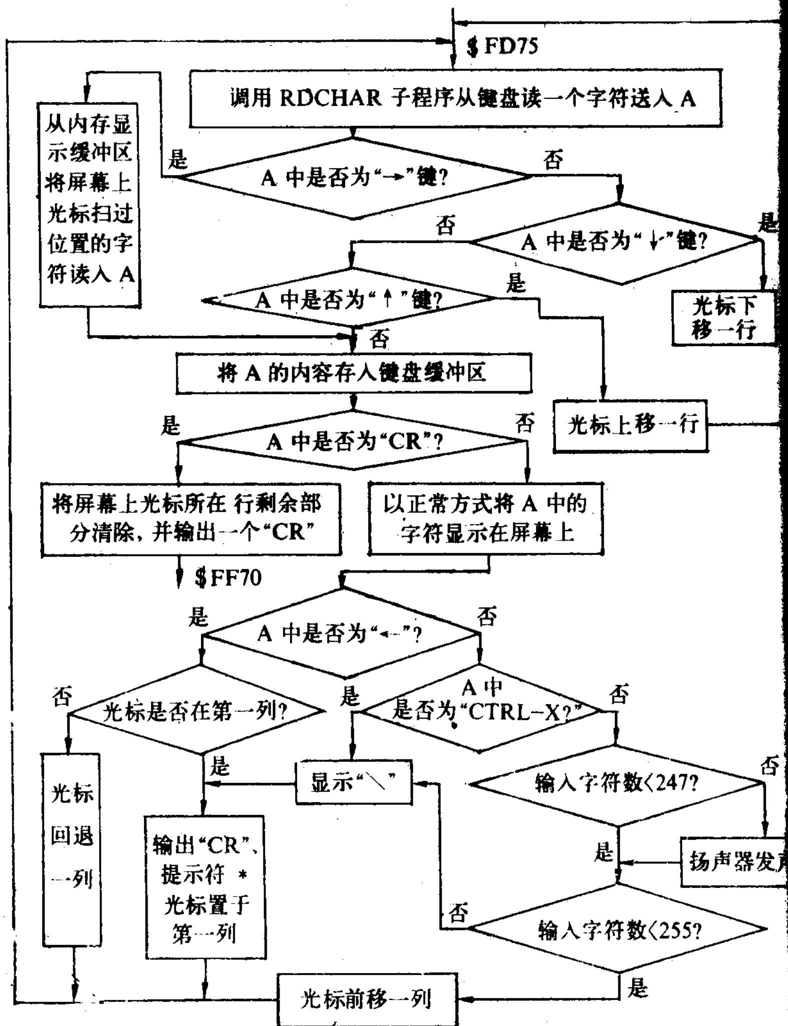


图2.3.6 监控命令的键入、保存与显示

3.2.2 读取监控命令

图 2.3.7 表示从键盘输入缓冲区读取监控命令的过程。

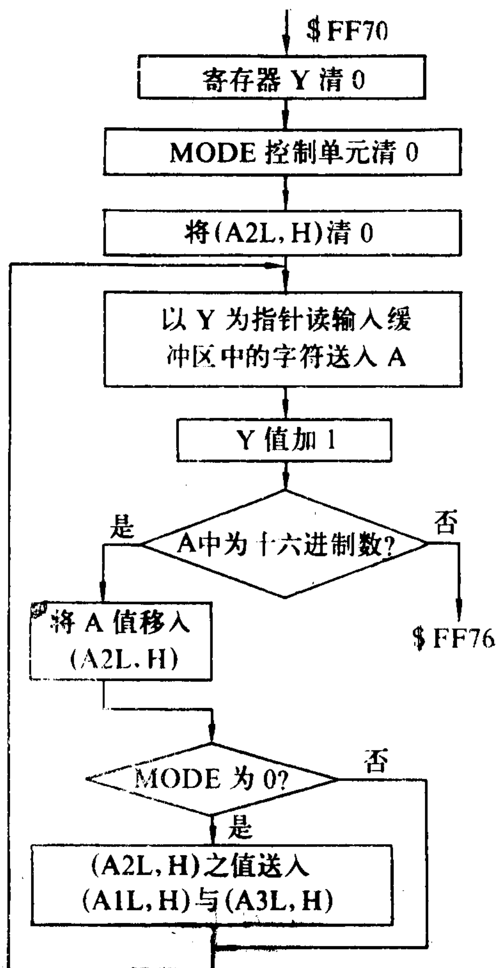


图 2.3.7 读取监控命令

3.3 监控中一些常用的子程序

3.3.1 RDCHAR 子程序

图 2.3.8 为 RDCHAR 子程序

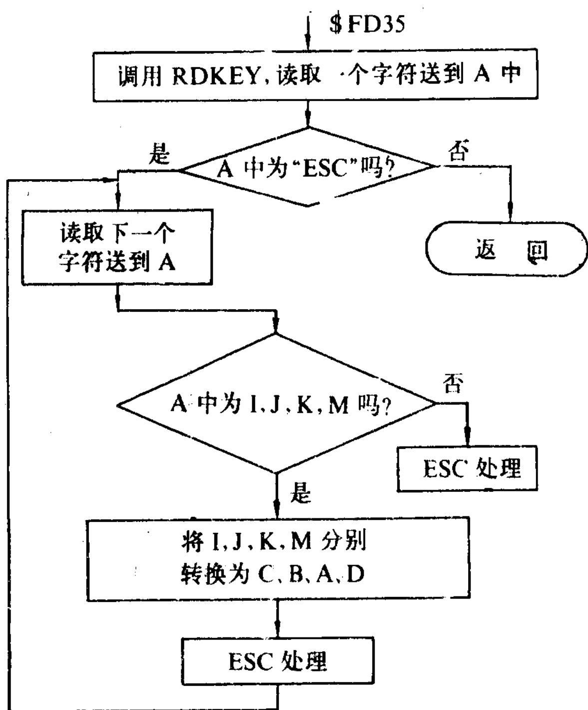
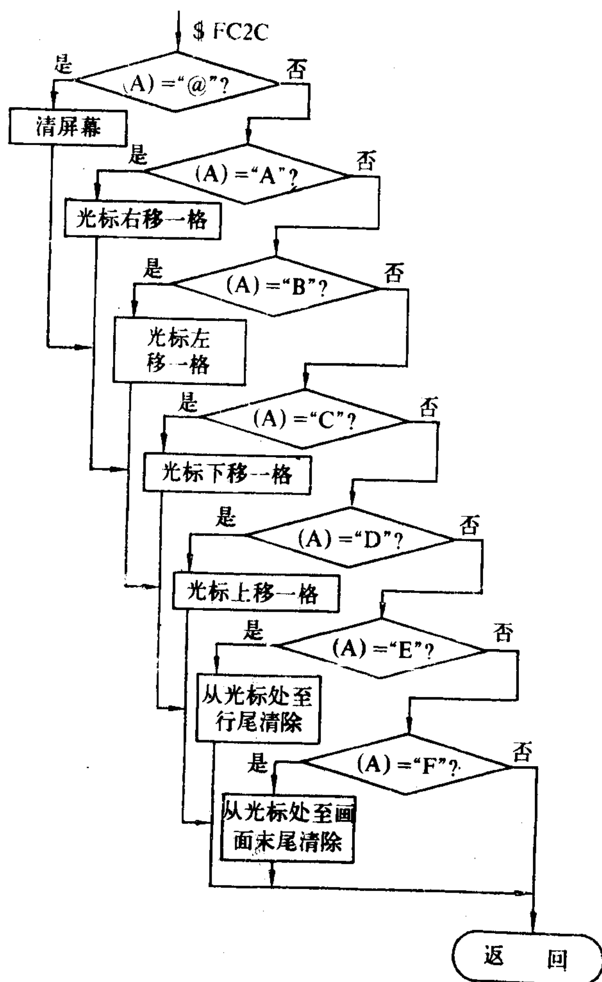


图 2.3.8 RDCHAR 子程序

3.3.2 ESC 处理

图 2.3.9 为 ESC 处理流程图。



$ FC2C
是
(A) ="@"?
否
清屏幕
是
(A) ="A"?
否
光标右移一格
是
(A) ="B"?
否
光标左
移一格
是
(A) ="C"?
否
光标下移一格
是
(A) ="D"?
否
光标上移一格
是
(A) ="E"?
否
从光标处至
行尾清除
是
(A) ="F"?
否
从光标处至画
面末尾清除
返 回

图 2.3.9 ESC 处理

3．禁止在开机状态拔插软盘驱动器和软盘驱动器接口卡。

4．不要在马达转动过程中抽、插软盘片。

5．控制温升变化和使用环境温度。温升变化范围应小于每小时 15 ℃、环境温度一般在 10 ℃～ 38 ℃之间。

6．避免灰尘的侵袭，不要将沾有灰尘的软盘片插入到软盘驱动器中。

7．磁头的清洗，可用专用的清洗软盘片或磁带清洗液。

8．软盘驱动器应远离磁场发生源，不用时不要将软盘片留在驱动器中。

1.3 软盘片

1.3.1 软盘片的包装与外形特点

软盘片密封于永久性保护套内(参见图 3.1.1)，盘片由圆形聚脂塑料做成。盘片表面涂有磁性材料。使用时，软盘片在保护套内旋转，磁头通过保护套中的长孔(又称磁头读／写窗口)和磁盘表面接触，读／写数据。

· 永久性保护套：由防静电皮革做成，内壁夹有防静电防潮材料，其作用是防止灰尘和保护盘面磁层不受损伤，同时也防止盘片旋转时所产生的静电而使信息丢失。

· 索引孔：用来进行盘片划分扇区的物理定位标志。由于大部分软盘驱动器中没有装上索引孔识别装置，故索引孔不起作用。

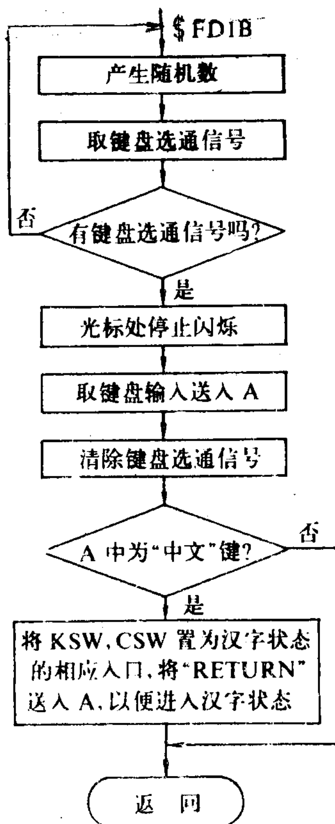


图 2.3.11 KEYIN 子程序

3.4 监控的其它功能

除了上面流程图所介绍的监控程序的功能外,它还具有

一些完成其它功能的子程序。如

IRQ / BRK 中断处理子程序;

低分辨率绘图子程序:

延时子程序;等等。

这些子程序的功能及接口信息,将在后面详细地介绍。

3.5 监控程序对第0页及第3页的使用

3.5.1 监控程序在第0页所使用的系统工作单元

表 2.3.1 及图 2.3.12 给出了监控程序在第 0 页中对系统工作单元的使用情况。

地址的	地址的低 4 位															
高 4 位	$0	$1	$2	$3	$4	$5	$6	$7	$8	$9	$A	$B	$C	$D	$E	$F
$0	•	•														
$1																
$2	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
$3	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
$4	•	•	•	•	•	•	•	•	•	•					•	•
$5																
$6																
$7																
$8																
$9																
$A																
$B																
$C																
$D																
$E																
$F																

图 2.3.12 监控程序对第 0 页单元的使用

表 2.3.1 第 0 页单元的监控程序符号表

地址	符号
$ 00	LOC0
$ 01	LOC1
$ 20	WNDLFT
$ 21	WNDWDTH
$ 22	WNDTOP
$ 23	WNDBTM
$ 24	CH
$ 25	CV
$ 26	GBASL
$ 27	GBASH
$ 28	BASL
$ 29	BASH
$ 2A	BAS2L
$ 2B	BAS2H
$ 2C	H2, LMNEM
$ 2D	V2, RMNEM
$ 2E	MASK, FORMAT, CHKSUM
$ 2F	LASTIN, LENGTH
$ 30	COLOR
$ 31	MODE
$ 32	INVFLG
$ 33	PROMPT
$ 34	YSAV
$ 35	YSAV1
$ 36	CSWL
$ 37	CSWH
$ 38	KSWL
$ 39	KSWH
$ 3A	PCL
$ 3B	PCH
$ 3C	A1L
$ 3D	A1H
$ 3E	A2L
$ 3F	A2H
$ 40	A3L
$ 41	A3H
$ 42	A4L
$ 43	A4H
$ 44	A5L
$ 45	A5H, ACC
$ 46	XREG
$ 47	YREG
$ 48	STATUS
$ 49	SPNT
$ 4E	RNDL
$ 4F	RNDH

3.5.2 监控程序对第 3 页单元的使用

监控程序仅仅把第 3 页的高端作为一些向量使用。其位置如下:

$ 03F0—$ 03F1: 这两个字节作为 BRK 指令的中断向量(地址)。

$ 03F2—$ 03F3: 这两个字节作为 RESET 向量(地址)。如果"$ A5"与 $ 03F3 的内容作异或的结果等于 $ 03F4 的内容,则 RESET 向量有效。否则,RESET 中断将使自启动监控程序做冷启动。

$ 03F8—$ 03FA: 这三个字节内通常放入一条 JMP 指令,作为 CTRL-Y 向量(指令)。

$ 03FB—$ 03FD: 这三个字节为不可屏蔽中断向量(指令)。

$ 03FE—$ 03FF: 此两字节为 IRQ 中断向量(地址)。

第四章 使用监控程序中的子程序

在监控程序中有许多可为其它程序调用的子程序。这些子程序将为其它程序的编制提供方便。下面将按这些子程序的功能分类介绍它们的入口地址、使用条件及执行结果。

入口地址以 16 进制地址 / 10 进制地址的格式写出。如果在监控程序中该子程序入口地址有标号,则将标号写在地址后面,在标号后面写出返主后被破坏的 CPU 寄存器的名

4.1 低分辨率绘图

F800 / 63488 / PLOT / A

功能:在行(A)列(Y)给出一个由(COLOR)决定色彩的低分辨率图形的点。返主后,GBASL,H 和 MASK 保留设定的状态。

F819 / 63513 / HLINE / A, Y

功能:在行(A)从左边(Y)列到右边(H2)划一条由(COLOR)决定色彩的低分辨率图形的水平线。

F828 / 63528 / VLINE / A

功能:在列(Y)从上边(A)到下边(V2)划一条由(COLOR)决定色彩的低分辨率图形的垂直线。

F832 / 63538 / CLRSCR / A, Y

功能:把低分辨率图形的 48 行全部清除为黑色(若在文字显示方式下,全屏幕将形成黑白相反的 @ 符号)。

F836 / 63542 / CLRTOP / A, Y

功能:将低分辨率图形上面的 40 行清除为黑色(若在文字显示方式下,上面 20 行字符变成黑白相反的 @ 符号)。

F838 / 63544 / CLRSC2 / A, Y

功能:将低分辨率图形的 0 行至(Y)行清除为黑色。

F83A / 63546 / A, Y

功能:将低分辨率图形的 0 行至(V2)行清除为黑色。

F847 / 63559 / A

功能:计算(A)所对应的内存基地址。(A)为低分辨率图形时屏幕的行号除 2 的值。对应该行的内存显示缓冲区基地

址被存入 GBASL,H。

F85F / 63583 / NXTCOL / A

功能:将现有颜色的彩色码加 3,设定为低分辨率图形的新颜色。

F864 / 63588 / SETCOL / A

功能:根据(A)中右半字节中的彩色值设定低分辨率图形的颜色。

F871 / 63601 / SCRN / A

功能:将行(A)列(Y)的低分辨率绘图点的色彩值取至 A 的右半字节。

4.2 输入

FD18 / 64792 / A

功能:间接转移至(KSWL,H)指向的输入子程序,通常是指向 KEYIN。

FD1B / 64795 / KEYIN / A

功能:等待键盘输入,形成随机数(RNDL,H),当测得有键盘输入时,将光标所指的闪烁字符恢复成正常显示,并将键盘输入数据读入 A。调用该子程序之前,光标所在行位置对应的内存显示缓冲区基地址应放入(BASL,H)中,光标列位置存入 Y 中,光标所在位置上的字符存于 A 中。返回调用程序后,A 中装有键盘输入的新数据。

FD0C / 64780 / RDKEY / A,Y

功能:把屏幕上光标位置的字符变成闪烁显示方式,而将原字符保留在 A 中,并经(KSWL,H)间接转移至输入子程序,通常是转移至 KEYIN 去读键盘。调用该子程序之前,光

标所在行位置对应的内存显示缓冲区基地址应放入(BASL, H),光标列位置在 CH 中,返回调用程序之后,A 中存有输入数据。

FD35 / 64821 / RDCHAR / A, Y

功能: 调用 RDKEY 子程序从输入设备读一个字符到 A 中。判断读入字符是否为 ESC。若是 ESC 键,则再调用 RDKEY 子程序,从输入设备读一个字符到 A 中,根据新输入的字符分别做如下处理:

@: 从行顶 WNDTOP 起清除屏幕显示的全部文字,返回 RDCHAR。

A: 光标向右移一列,返回 RDCHAR。

B: 光标向左移一列,返回 RDCHAR。

C: 光标向下移一行,返回 RDCHAR。

D: 光标向上移一行,返回 RDCHAR。

E: 从光标处清除至本行末尾,返回 RDCHAR。

F: 从光标处清除至画面末尾,返回 RDCHAR。

I、J、K、M: 分别变换成 D、B、A、C,做相应处理后,再调用 RDKEY 子程序,从输入设备再读一个字符至 A 中,然后根据该字符再做上述处理。

若不是 ESC 键,则直接返回调用程序(参见图 2.3.8 和 2.3.9)。

FD67 / 64871 / GETLNZ / A, X, Y

功能: 输出一个 RETURN 及 PROMPT 中的提示符之后,从输入设备连续接收一串以 RETURN 为结尾的字符(最多允许 254 个),存入由 $ 200 开始的输入缓冲区,并将这一串字符显示在屏幕上(参见图 2.3.6)。

4.3 屏幕的清除与光标的控制

FC42 / 64578 / CLREOP / A, Y

功能: 将文字显示屏幕从光标所在位置行(CV)列(CH)处开始清除到卷动画面的结尾处。

FC46 / 64582 / CLEOP1 / A, Y

功能: 将文字显示屏幕从行(A)列(Y)处开始清除到卷动画面的结尾处。

FC58 / 64600 / HOME / A, Y

功能: 从行顶 WNDTOP 起清除屏幕显示的全部文字, 光标设定在屏幕的左上角。

FC95 / 64661 / SCRL3 / A, Y

功能: 清除 (BASL, H) 对应的一行文字显示, 然后用 CV 和 WNDLFT 来设定 BASL, H, 即光标放在(CV)行的左端。

FC9C / 64668 / CLREOL / A, Y

功能: 将文字显示从光标位置开始清除至行末。在调用该子程序前应将要清除的行位置所对应的内存显示缓冲区基地址放入到 BASL, H 中。

FC9E / 64670 / CLEOLZ / A, Y

功能: 将文字显示从 (BASL, H) + (Y) 处起清除到行末。

FC10 / 64528 / BS / A

功能: 若光标不在右端, 则左移一列; 若光标在左端且在行顶, 则移到本行的右端; 若光标在左端且不在行顶, 则移到上一行的右端, 并按新的行位置更新 BASL, H。

FC1A / 64538 / UP / A

功能：如果光标不在行顶，即(CV) < (WNDTOP)，则光标上移一行，并按新行位置更新 BASL，H；若光标在行顶，则行位置不变。

FC62 / 64610 / CR / A，Y

功能：若光标不在底行，则光标移至下一行左端，并按新行位置更新 BASL，H；若光标已在底行，则光标移至该行左端，画面向上卷动一行，清除底行。

FC66 / 64614 / LF / A，Y

功能：若光标不在底行，则不改变水平位置，垂直下移一行，并按新行位置更新 BASL，H；若光标已在底行，则行位置不变，画面向上卷动一行，并将底行清除。

FBF4 / 64500 / ADVANCE / A，Y

功能：若光标不在右端，则光标右移一列；若光标在右端但不在底行，则光标移至下一行左端，并按新行位置更新 BASL，H；若光标在屏幕右下角，则光标从底行左端到右端，画面向上卷动一行，并清除底行。

4.4 文字显示及其内存缓冲区地址计算

FB2F / 64303 / INIT / A

功能：将 CRT 设定为以 24 行 40 列的格式显示第一页显示缓冲区内的文字，将光标置于底行，并将 STATUS 清零。

FB5B / 64347 / TABV / A

功能：将行位置(A)存入 CV 后，转到 VTAB 去计算 BASL，H

FBC1 / 64449 / BASCALC / A

功能:计算行位置(A)对应的第一页文字缓冲区的基地址,放入 BASL,H 中。

FC22 / 64546 / VTAB / A

功能:根据行位置(CV)及字幕左边界(WNDLFT),计算该行左边界对应的第一页文字显示缓冲区地址,放入 BASL,H 中。

FC24 / 64548 / VTABZ / A

功能:根据行位置(A)及字幕左边界(WNDLFT),计算该行左边界对应的第一页文字显示缓冲区地址,放入 BASL,H 中。

FC70 / 64624 / SCROLL / A,Y

功能:将文字显示屏幕向上卷动一行。

FE80 / 65152 / SETINV / Y

功能:将文字显示方式设定为反相方式,即白底黑字。

FE84 / 65156 / SETNORM / Y

功能:将文字显示方式设定为正常方式,即黑底白字。

4.5 输出

F940 / 63808 / PRNTYX / A

功能:将(Y)(X)用四位 16 进制数的 ASCII 码送到被选定的输出设备上。

F941 / 63809 / PRNTAX / A

功能:将(A)(X)用四位 16 进制数的 ASCII 码送到被选定的输出设备。

F944 / 63812 / PRNTX / A

功能:将(X)以两个 16 进制数的 ASCII 码,送到选定的

输出设备上。

F948 / 63816 / PRBLNK / A,X

功能:送出三个空格到选定的输出设备上。

F94A / 63818 / PRBL2 / A,X

功能:送出(X)个空格到选定的输出设备上。

FB78 / 64376 / VIDWAIT / A,Y

功能:在允许操作者要求暂停的情况下,处理累加器A中的控制字符或在光标位置显示累加器A中的显示字符。进入该子程序时,若累加器A中是"RETURN"字符,且键盘输入为"CTRL-S",则清除键盘选通信号后,进入等待键盘重新输入状态,直至有新键入字符为止。若新键入的字符是"CTRL-C",则保留该键盘字符转入VIDOUT;否则,清除键盘选通信号后转入VIDOUT。若累加器A中不是"RETURN"字符或键盘没输入"CTRL-C",则直接转入VIDOUT去执行。

FBFD / 64509 / VIDOUT / A,Y

功能:若累加器A中的字符是可显示字符,则写入光标所对应的内存显示缓冲区,然后,若光标不在左端,则光标右移一列,否则移至下一行行首。若A中不为可显示字符,则做相应的控制动作($ 8D转至CR; $ 8A转至LF; $ 88转至BS;否则使扬声器发声)。调用该子程序之前,光标列位置已存入CH,光标行位置左端对应的内存显示缓冲区地址已存入BASL,H(参见图2.3.10)。

FD8B / 64907 / A,Y

功能:从光标所在位置到该行右端清屏幕,然后输出一个"RETURN"。

FD8E / 64910 / CROUT / A

功能：将“RETURN”输出到选定的输出设备上。

FDDA／64986／PRBYTE／A

功能：将(A)以两个16进制数的ASCII码送到选定的输出设备上。

FDE3／64995／PRHEX／A

功能：将(A)的低半字节中的16进制数变成ASCII码送到选定的输出设备上。

FDED／65005／COUT

功能：由(CSWL，H)指向的输出子程序，将累加器A中的字符输出到一个选定的输出设备上。一般是由COUT1将A中的字符在CRT上显示或处理。

FDF0／65008／COUT1

功能：将累加器A中的显示字符，经INVFLG处理成正常、反相或闪烁字符，再经由VIDWAIT和VIDOUT显示在光标位置上；若累加器A中是控制字符，则不经INVFLG处理，直接经由VIDWAIT和VIDOUT做适当的控制处理。

FF2D／65325／PRERR／A

功能：送出“ERR”到选定的输出设备上，并使扬声器发声。

4.6 监控命令

FE2C／65068／MOVE／A，Y

功能：将内存中A1L，H所指地址到A2L，H所指地址范围内的数据，传送至以A4L，H所指地址为起始地址的地方。调用前应将Y寄存器置为0。

FE36／65078／VFY／A，X，Y

功能:将内存中 A1L,H 所指地址到 A2L,H 所指地址范围内的数据,与 A4L,H 所指地址为起始地址的数据块进行逐个比较,若有不相等,则将不相同的数据及其地址显示出来。调用前应将 Y 寄存器置为 0。

FE5E / 65118 / LIST / A,X,Y

功能:将 A1L,H 所指地址开始的 20 条机器语言指令的反汇编符号显示出来。

FEB6 / 65222 / GO / A,X,Y

功能:转入 A1L,H 所指地址处执行。

FECD / 65245 / WRITE / A,X,Y

功能:将 A1L,H 所指地址至 A2L,H 所指地址之间的数据写至磁带上。

FED0 / 65248 / TRACE / A,X,Y

功能:跟踪执行从 A1L,H 所指地址开始的程序,每执行一条指令后,将各寄存器之值及指令的反汇编码显示出来。

FED2 / 65250 / STEPZ / A,X,Y

功能:单步执行 A1L,H 所指地址处的机器指令,并将执行后各寄存器之值及指令的反汇编码显示出来。执行完毕后,将下一条指令的地址置入 A1L,H 中。

FEFD / 65277 / READ / A,X,Y

功能:将磁带上的内容写至 A1L,H 所指地址与 A2L,H 所指地址之间的存储区中。磁带数据长度应该与该区域的长度相等,否则,给出错误信息。

4.7 其　它

FB1E / 64286 / PREAD / A,Y

功能: 读游戏控制器上第(X)号模拟量输入,返回时结果存于 Y 中。

FBD9 / 64473 / BELL1 / A, Y

功能: 若(A) = $ 87(即"BELL"),则等待 0.01 秒后扬声器产生 1kHz 声音,持续时间为 0.1 秒。若(A)≠ $ 87,则返回。

FBE2 / 64482 / A, Y

功能: 扬声器产生 1kHz 的声音,持续 0.1 秒。

FBE4 / 64484 / BELL2 / A, Y

功能: 扬声器产生 1kHz 的声音,持续时间由(Y)决定。

FCA8 / 64680 / WAIT / A

功能: 使得计算机延迟一段时间。累加器 A 的值决定调用该子程序的延迟时间:

延迟时间 =$(2.5 * A^2 + 13.5 * A + 13) * 1.023\mu s$

FEB0 / 65200 / XBASIC

功能: 进入 BASIC 语言,并清除存储器中的程序与变量。

FEB3 / 65203 / BASCOUNT

功能: 返回 BASIC 语言,并保留存储器中的程序与变量。

FF3A / 65338 / BELL / A

功能: 扬声器产生 1kHz 的声音,持续 0.1 秒。

FF3F / 65343 / RESTORE / A, X, Y, P

功能: 从零页系统工作单元 $ 45 ~ $ 48 恢复累加器 A,变址寄存器 X, Y,状态标志寄存器 P。

FF4A / 65354 / SAVE / A, X

功能: 将累加器 A,变址寄存器 X, Y,状态标志寄存器 P.

堆栈指示器 S 保存到内存零页系统工作单元 $ 45～$ 49，并清除 10 进制工作方式。

FF69／65385／MONZ

功能：进入监控命令处理器。

第五章　小汇编程序

小汇编程序是中华学习机固化在 ROM 中的一个工具软件，它的作用是将 6502 汇编记忆符翻译成 6502 机器指令。

5.1　小汇编的进入及退出

在监控状态下敲入下列命令行，即可进入小汇编状态：

* D350G

这时，屏幕显示小汇编状态的提示符"！"，表示系统已进入小汇编状态。

当需要退出小汇编状态时，只需敲入下列命令行：

*　$ D360G

5.2　汇编过程

在小汇编提示符下，可按下列格式进行工作：

〈地址〉：〈汇编记忆符〉

每敲入一行上述格式的命令，便汇编一条 6502 汇编指令，即将对应的机器指令存入冒号（：）前面的地址（若为多字

节指令，则存入以该地址为首地址的连续几个存储单元)。

若下一条指令接着上一条指令存放，则可省略地址的键入，这时，只需在空格符后直接敲入汇编记忆符即可。

5.3 使用监控命令

在小汇编状态下，可以使用各种监控命令，使用的方法是在各监控命令的前面加上符号“$”。

5.4 注意事项

由于中西文状态在存储器空间上的冲突，小汇编程序只宜在西文状态下使用。

第三部分
DOS磁盘操作系统

第一章 引 言

中华学习机(CEC-2000型)是与紫金Ⅱ/APPLE Ⅱ相兼容的家庭学习电脑,其硬件、软件与APPLE Ⅱ完全兼容。它具有软盘驱动器接口控制电路,可以直接和软盘驱动器相联接,可以运行在紫金Ⅱ/APPLE Ⅱ机上开发的各种磁盘软件。

中华学习机有丰富的系统软件和应用软件,它可以运行多种磁盘操作系统及各操作系统所支持的各种语言和应用软件、教学软件。它可以运行的磁盘操作系统包括有:APPLE DOS、PRODOS、UCSD-P和CP/M操作系统等等,其基本用法和APPLE Ⅱ机完全相同。

在这众多的系统软件和应用软件中,用户接触最多、使用最广泛的是APPLE DOS磁盘操作系统。本文仅以此为基础,向用户介绍APPLE DOS操作系统的基本命令及其用法。

1.1 APPLE DOS磁盘操作系统

APPLE DOS 磁盘操作系统(简称为:APPLE DOS 或 DOS 操作系统)是美国 APPLE 计算机公司为 APPLE Ⅱ微机系统的使用而设计开发的一种磁盘操作系统。它面向磁盘机的使用和磁盘文件的管理,具有设计简单、面向用户、使用方便的特点。

APPLE DOS 操作系统的开发和完善大致经历了三个主要阶段:

1. 诞生阶段:1978 年 6 月 29 日和 7 月 20 日分别推出了 DOS 3.0 和 DOS 3.1 版本。当时 DISK Ⅱ子系统的软盘读 / 写格式为 35 道、13 扇区,磁盘最大容量为 114 KB。

2. 维护改进阶段:1979 年 4 月 16 日和 7 月 31 日又先后推出 DOS 3.2 和 DOS 3.2.1 版本。这二个版本的软件是在 DOS 3.1 基础上对其进行纠错和改进。

3. 更新阶段:1980 年 8 月 25 日推出全新的 DOS 3.3 版本。这是从硬件、软件二个方面同时对 DOS 3.2.1 版本进行全面改进的结果。DISK Ⅱ子系统的软盘读写格式由 13 扇区改成 16 扇区,磁盘容量达到 143 KB。软件上增加了存储文件过程中自动进行校验等功能,为今后的 DOS 3.4、RAM DISK 及改进型 DOS 3.3 版本的开发提供了良好的开发环境。

为了能兼容于 DOS 3.2.1 以前版本软件的使用,在 DOS 3.3 系统主盘上向用户提供了 13 扇区到 16 扇区的转换程序、13 扇区磁盘的引导程序等。

APPLE DOS 操作系统除了包括了一般磁盘操作系统所

具有的基本功能(即磁盘数据的读 / 写、磁盘文件的建立和管理、输入 / 输出接口命令等等)外,还专门为 BASIC 程序的使用提供了更多的操作命令。它与系统软件、用户程序的接口界面非常清晰,用户可以通过它所提供的一些命令很容易实现主机内存与磁盘之间的信息交换。

1.2 软盘驱动器

1.2.1 磁盘子系统

磁盘驱动器和显示器、录音机一样,是微电脑系统中一个重要的外部设备。利用磁盘驱动器这样一个外存储设备,可以将主机内存的程序和数据记录到磁盘上,并可以从磁盘上读回,达到重复使用、修改的目的。

磁盘驱动器、磁盘控制驱动电路以及与主机相连接的接口构成了微电脑系统的磁盘子系统。磁盘子系统为磁盘操作系统软件的使用提供了硬件环境的支持。

由于配置的磁盘驱动器的类型规格不同,相应的盘控驱动电路、控制接口电路也可能完全不同,它们所构成的磁盘子系统也就不同。目前,中华学习机(CEC-2000 型)所具有的盘驱动器接口控制电路只支持 5.25 英寸单面单密度软盘驱动器的使用,接口控制信号与 APPLE DISK Ⅱ接口控制信号相兼容。

1.2.2 软盘驱动器是如何工作的

软盘驱动器和磁带录音机一样,都是利用磁化方式来储存数据的,软盘驱动器的存储介质是软盘片,它与磁带之间的

主要差别就在于软盘片如同唱片一样，可以在软盘驱动器中进行旋转。而软盘驱动器内部的读 / 写磁头在其定位装置的控制下，可以在磁盘表面的磁道上进行移动，磁头下所经过磁盘表面存储介质就会被磁化，读 / 写数据就会记录在磁盘上。

由于磁盘的磁轨由同心圆组成，起始点和终止点相同，这样软盘驱动器就可以随机地读 / 写软盘上的信息，因此，它要比磁带机的顺序查找、读 / 写数据的方式快得多，也准确得多。

1.2.3 单盘驱动系统和多盘驱动系统

一台主机根据它所带有的软盘驱动器控制接口数可以配接一台或一台以上的软盘驱动器。由一台盘驱构成的系统我们称为单盘驱系统，而由二台以上盘驱构成的系统我们称为多盘驱系统。

中华学习机(CEC-2000型)仅有一个软盘驱动器控制接口，它可以通过20线偏平电缆与一台软盘驱动器相连接。第一台驱动器在DOS操作系统下的设备号为：6＃槽口，1＃驱动器。记为：S6，D1。

对于中华学习机，由于受扩充槽口数的限制，最多可接3台软盘驱动器。

1.2.4 软盘驱动器使用注意事项

1. 使用软盘驱动器要轻拿轻放，避免剧烈震动，以防定位机构松动。

2. 避免在软盘驱动器工作时移动盘驱本身。

3.3.3 VIDOUT 子程序

VIDOUT 子程序如图 2.3.10 所示。

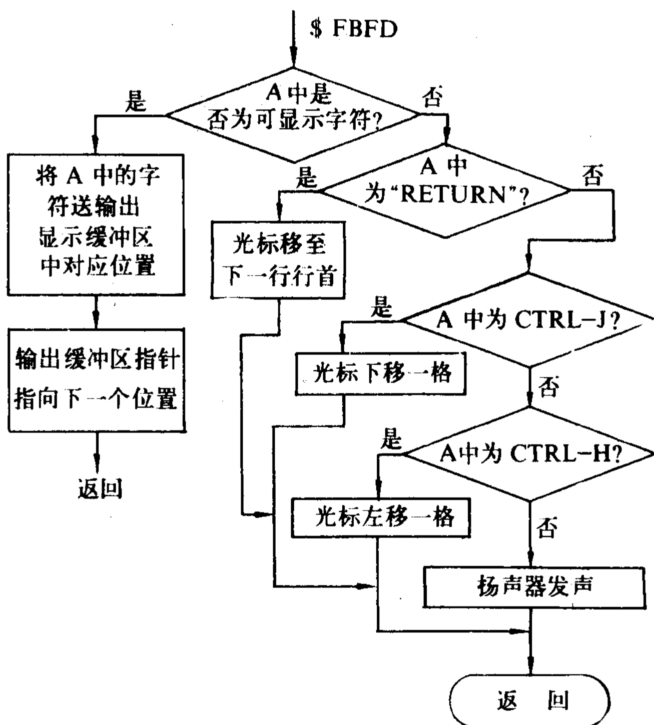


图 2.3.10 VIDOUT 子程序

3.3.4 KEYIN 子程序

KEYIN 子程序如图 2.3.11 所示。

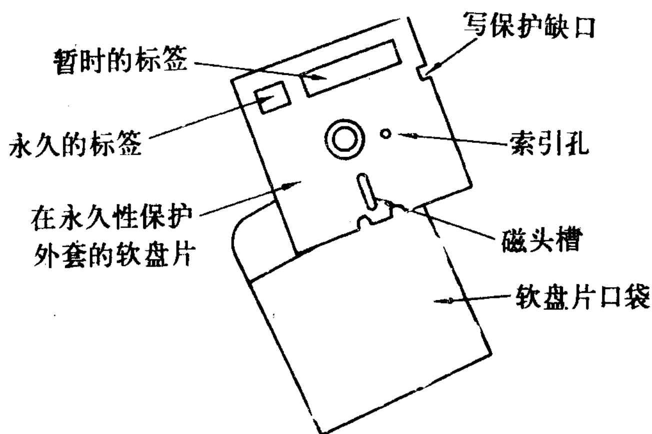


图 3.1.1

• 磁头读／写窗口：这是在永久性保护套上开辟的一个长孔，是磁头与软盘片接触，读／写信息的界面。因此，在软盘片的使用过程中，绝对禁止用手指或异物接触这个长孔中所暴露出的磁盘部分。

• 写保护切口：控制着软盘驱动器写电路开关，当用写保护封条贴在该切口上时，可以防止磁盘被写入任何信息。

· 永久性标签: 出厂时贴上的磁盘说明。用以指明磁盘的牌号和盘片的型号等。在使用盘片时,我们最好用手指捏着这部分取盘或插盘。

· 临时标签: 在使用软盘时,我们希望能够通过简单的方法注明该盘片的名称、作用等。此时我们可以用不干胶标签贴上,用软笔写上名称、标记等。不需要时,可以撕该标签,换贴新的标签。

· 盘纸套: 用来保护盘片,防止灰尘的侵蚀。我们要养成把不用的盘片及时插入盘片纸套中的习惯。

1.3.2 软盘片的选择

软盘片有多种型号和规格,它适用于不同类型的软盘驱动器。按照软盘驱动器磁头工作方式可以分成单面和双面读写,按照数据的记录方式又有单密度和双密度二种,按照扇区划分又有硬分段和软分段之分。

对于中华学习机和 APPLE Ⅱ来说,选用的是 5.25 英寸单面单密度软盘驱动器。因此,它对软盘片的选择要求比较低。一般地说,单面单密度、单面双密度、双面单密度和双面双密度的 5.25 英寸软分段软盘片均适合其使用。

1.3.3 磁道和扇区

软盘驱动器在读写信息过程中,由于盘片的旋转,读 / 写磁头将把盘片划分成若干个同心圆。这些同心圆称为磁道,如图 3.1.2 所示。

读 / 写磁头通过软盘的读 / 写窗口,可以从一个磁道移到另一个磁道,这可使磁头找到要读写的数据区。

软盘上的数据是以数据块的形式存放在磁道上的,每一个数据块称为一个扇区。

在 APPLE DOS 3.3 操作系统支持下,每张软盘被划

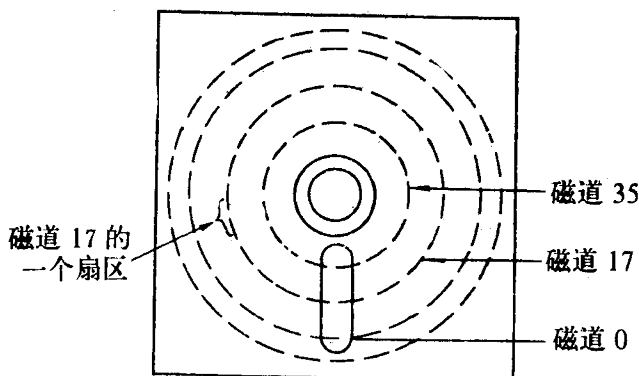


图 3.1.2

分为 35 个磁道，每个磁道上有 16 个扇区，每个扇区可存放 256 个字节的信息(不包括数据块的地址索引)。因此，每张盘片的最大存储容量为 140 KB 左右。

1.3.4 软盘片使用注意事项

1. 软盘片不能受重压，不可弯曲，使用后要及时插入盘片的纸盘套内，立放于软盘盒中。

2. 使用时把软盘片有标签的面朝上，用右手拇指按着有永久性标签处，将有磁头读 / 写窗口的一边先插入

驱动器，用拇指慢慢推入软盘片，直到盘片全部进入驱动器。

3. 取出时用食指和拇指捏着盘片的边缘，反方向抽出。

4. 不要在驱动器旋转和亮灯情况下，打开驱动器门，移出软盘片。这样容易损坏软盘片，丢失盘片上的信息。

5. 使用中要注意保持盘片的清洁，避免灰尘和油污的污染，绝对禁止用手触摸暴露在磁头读 / 写窗口和索引孔中的磁盘部分。

6. 盘片的保存温度最好在 10 ℃— 50 ℃之间，要避开磁场的干扰，保持干燥。

第 二 章
DOS 操作系统的引导和使用

2.1 磁盘操作系统的引导

无论是 DOS 3.3 操作系统，还是 PRODOS 操作系统，或者是 UCSD–P 操作系统，其系统程序都是以某种数据形式存储在软盘片上的。如果想使用其中某一磁盘操作系统进行工作，首先遇到的问题是如何将该操作系统程序从软盘上装入到主机内存中来，然后才是怎样利用该操作系统所提供的命令进行操作。这种将操作系统装入内存，并使其工作的过程称为磁盘操作系统的引导。

在中华学习机上可以有两种方式来进行磁盘操作系统的引导——开机引导和命令方式引导。

2.1.1 开机引导

中华学习机固化软件中有一个自启动监控程序，开机后机器首先执行的是这段程序。这个程序会从与主机直接相连接的软盘驱动器(设备号为：S6，D1)中读磁盘，装入磁盘操作系统。因此，只要用户在开机前装好软盘驱动器，并将操作系统盘插入软盘驱动器，然后，打开主机电源开关，机器便会自动装入操作系统。

这种开机引导磁盘操作系统的方式称为“冷启动”方式或“自启动”方式。

2.1.2 命令方式引导

如果主机已加电运行，需要装入或重新装入磁盘操作系统时，可以将操作系统盘插入软盘驱动器，根据下列所述状态，打入相应命令，即可进行操作系统的引导。

系统状态	提示符	打入命令
APPLESOFT	]	PR #6 或 IN #6
整数 BASIC	>	PR $ 6 或 IN #6
监 控 程 序	*	6 CTRL--P 或 C600G (注)
小 汇 编	!	$ C600G

这种在主机已加电情况下，引导操作系统的方法又称为“热启动”。

(注)“CTRL-P”表示在键盘上按下 CTRL 键的同时，按下 P 键。“CTRL-A”等类似。

2.2 用DOS 3.3系统主盘进行启动

现在我们可以尝试一下如何用 DOS 3.3 系统主盘来引导 DOS 操作系统的方法。

1．从随机附带的软盘片盒中，取出一张贴有“DOS 3.3 系统主盘”标签的软盘片。

2．插入与主机相连接的软盘驱动器中，关上驱动器的门。

3．打开主机电源开关。

4．此时软盘驱动器上的红色指示灯亮，磁盘同时转动起来。

5．软盘驱动器指示灯熄灭后，屏幕将显示如下信息：

DOS VERSION 3.3　　　　　08／25／80
APPLE II PLUS OR ROMCARD　SYSTEM MASTER
(LOADING INTEGER INTO LANGUAGE CARD)
]

至此，启动执行完毕，DOS 操作系统已装入内存，并准备接受命令。

做完了冷启动操作，如果我们还想再做一次“热启动”，那么，你打入命令 PR #6 或 IN #6，机器重复做上面所说的第 4，第 5 项操作，于是你又做了一次“执启动”操作。

如果你还想尝试一下在整数 BASIC 状态下进行“执启动”是怎么一回事，请你打入如下命令：

]INT ✓ (注)

>PR #6✓

2.3 初始化新盘片

新买来的空白软盘片，上面没有记录任何信息，在使用DOS命令对空白片记录信息以前，必须对它们进行格式化，格式化的目的是在磁盘上按磁道和扇区来划分地址，并把地址以数据的形式记录在软盘上，便于读 / 写数据时，按其地址进行查找。

在DOS 3.3操作系统命令中，没有单独提供一条格式化磁盘的命令，但它提供了一条初始化盘片的命令(INIT)。利用初始化操作命令，可以完成对软盘进行格式化的操作。

初始化操作的步骤：

1. 用DOS系统盘进行启动。

2. 从盘上装入一个BASIC程序，或从键盘上键入一个新的BASIC程序。

3. 移去系统盘，插入空白盘，并关上驱动器门。

4. 打入命令：INIT 文件名 ✓

当屏幕上再次出现输入提示光标后，初始化操作执行完毕。

现在我们来做一下初始化新盘片的操作：

1. 取出DOS 3.3系统主盘进行启动。

2. 出现输入行提示光标后，打入如下命令：

]NEW

]10 TEXT：HOME

(注) ✓ 表示按下RETURN键。

```
]20 VTAB 10:HTAB 7:PRINT "THIS DISKETTE
CREATED BY"
]30 VTAB 12:HTAB 7:PRINT "WANG PIN
10/08/87"
]40 VTAB 20
]50 END
```

3. 取出 DOS 3.3 系统主盘,并插入空白盘片。

4. 打入初始化命令:

```
]INIT GREETING
```

这样就可以完成对空白盘片的初始化操作。

在 APPLE DOS 操作系统中,所有经过初始化操作的空白盘片都存有 DOS 磁盘操作系统程序,也都可以作为操作系统的引导盘进行启动。

现在,我们不要移去新初始化后的盘片,关掉主机电源开关后数秒钟,再重新打开主机电源开关,屏幕上将出现如下信息:

```
THIS DISKETTE CREATED BY
WANG PIN          10/08/87
]
```

这表明:我们用新初始化后的盘片引导 DOS 3.3 操作系统成功。这个盘片我们下面还要用到,权且称它为"工作盘"。

2.4 列文件目录

我们在使用软盘片过程中,很希望知道当前的软盘片已存放着哪些文件,列磁盘文件目录命令可以帮助我们完成这一工作。

现在,我们打入命令:CATALOG,屏幕上将出现如下信

息:

DISK VOLUME 254

A 002 GREETING

如果我们换上 DOS 3.3 系统主盘，再打入 CATALOG 命令，屏幕上将出现如下信息:

```
DISK VOLUME 254 ——————————————— ①
 * A 008 HELLO
 * I 018 ANIMALS
   T 003 APPLE PROMS
 * I 006 APPLESOFT
 * I 026 APPLEVISION
 * I 017 BIORHYTHM
 * B 010 BOOT13
 * A 006 BRIAN'S THEME
 * B 003 CHAIN
 * I 009 COLOR DEMO
 * A 009 COLOR DEMOSOFT
 * I 009 COPY
 * B 003 COPY.OBJ0
 * A 009 COPYA
 * A 010 EXEC DEMO
 * B 020 FID
 * B 050 FPBASIC
 * B 050 INTBASIC
 | |   |    |
 | |   |    └─────────────────── ②
 | |   └──────────────────────── ③
 | └──────────────────────────── ④
 └────────────────────────────── ⑤
```

通过屏幕上所列出的这些信息，我们可以知道以下几件事情：

① 该盘片所占有的卷号。一张盘片只能有一个卷号，它是在初始化命令执行过程中所确定的。

② 文件名，每个文件名最大长度为 30 个字符。

③ 文件在软盘上所占用的扇区数。

④ 文件类型：

A 表示 APPLESOFT 程序文件。

I 表示整数 BASIC 程序文件。

B 表示 2 进制代码文件。

T 表示文本文件。

R 表示由 EDASM 汇编程序所产生的可浮动目标文件。

L 表示由 LISA 汇编程序所生成的文件。

⑤ 文件保护标志。它是操作系统所提供的对文件进行写保护的一项措施。

如果该文件有星号标志(*)，则表示 DOS 对该文件进行了封锁保护，它只允许用户对该文件执行读操作，而不能对其执行改名、删除或写操作。否则，系统会给出：FILE LOCKED 信息。

在列目录过程中，可以用 CTRL-S 键暂停列目录，再按任意键使其继续列目录。如果目录内容多于屏幕一次能够显示的条目时，可按任意键使其继续显示。

2.5 保存 BASIC 程序

当我们在主机上编制了一个 BASIC 程序，很希望把它保

存到软盘上去，在下次上机时，随时把它调入到主机的内存中来。这时，我们就可以使用 DOS 操作系统所提供给我们的存盘命令：SAVE。

在这个 DOS 命令中，我们要给出该程序在软盘上的文件名。这个文件名是以字母打头的字符串（但不能含有逗号），文件名的最大长度为 30 个字符。

例如：我们在主机上打入 BASIC 程序：

```
10  INPUT "GUESS A NUMBER: ";G
20  READ D
30  IF D = - 99999 THEN 80
40  IF G < > D  THEN 20
50  PRINT "YOU ARE  CORRECT !"
60  END
80  PRINT "BAD GUSS,  TRY AGAIN !":
       RESTORE : GOTO 10
100 DATA 1,393,-39,28,391,0,3.14
       ,90,15,10,5,-34,-99999
```

打入 RUN 命令检查程序执行正确后，可以把该程序保存到工作盘上，我们把该程序取名为：GUESS A NUMBER。可以打入如下命令：

```
]SAVE GUESS A NUMBER
```

此时，若打入列文件目录命令，我们将会看到工作盘上已含有二个文件：

```
]CATALOG

DISK VOLUME 254

A 002 GREETING
```

A 002 GUESS A NUMBER

利用 SAVE 命令，我们可以在工作盘上保存多个程序文本，换上其它软盘，还可以把这个程序保存到其它软盘上。

在使用 SAVE 命令时，我们要注意：保存程序的软盘片不能贴有写保护，也不能是没有经过初始化的空白盘片。否则，我们会得到信息：I／O ERROR。

2.6 装入 BASIC 程序

启动 DOS 操作系统后，我们可以利用 DOS 命令把软盘上的程序装入到主机内存中来。

例如：我们希望把在工作盘上的 GUESS A NUMBER 程序装入到主机内存，则可以打入如下命令：

]LOAD GUESS A NUMBER

在用 LOAD 命令装入程序时，主机内原有的 BASIC 程序将被清除掉，由新装入的程序替换原有的程序，而软盘上的程序文件不会受任何影响。

如果用 LOAD 命令所指定的文件名不在软盘上，将得到信息：FILE NOT FOUND。

2.7 删除文件

在我们使用软盘来存储信息时，很希望在盘上保留对我们有用的信息，对于调试程序中留下的不用的旧版本程序，旧的数据文件，我们都可以用删除命令把它们删除，以使软盘上可以留下更多的空间来存放新的程序、新的数据。

例如：我们希望把工作盘上的 GUESS A NUMBER 删

除掉，可以打入如下命令：

]DELETE GUESS A NUMBER

要注意：打入 DELETE 命令后，软盘上所指定的文件名将被从文件目录中删除掉，并且是不可重新恢复的，所以操作时要十分小心，只有当我们证实了该文件确实对我们无用时，才可将该文件删除掉。在一般情况下为了防止因我们错误的操作而将盘上重要的文件删除掉，我们将对这些文件用 LOCK 命令加写封锁标志。

如果在 DELETE 命令中所指定的文件名不在软盘上，将得到信息：FILE NOT FOUND。

如果软盘上贴有写保护，将得到信息：

WRITE PROTECTED

如果 DELETE 命令中所指定的文件名，是用 LOCK 命令加了封锁的，将得到信息：FILE LOCKED。

如果我们用 LOAD 命令，而要装入的文件不是 BASIC 程序(即文件类型不是 A 或 I)，则得到信息：

FILE TYPE MISMATCH

第 三 章
DOS 3.3 磁盘操作系统命令

3.1 DOS 命令格式

DOS 操作系统命令由命令保留字和命令参数二部分组

成。

命令保留字代表了 DOS 操作系统所提供的命令。它必须由大写字母组成。

命令参数则是提供 DOS 命令执行时所需要的辅助信息。如:磁盘驱动器的设备号,文件存放的地址,磁盘的卷号,,文件名等等。命令参数由文件名、参数标记和参数组成。参数标记必须是大写字母,参数则可以是算术表达式。

一条 DOS 命令可以带有若干命令参数,命令参数之间必须用逗号隔开。

DOS 命令格式如下:

COMMAND f [,Ss][,Dd][,Vv]

命令保留字　　　　命令参数

在上述 DOS 命令语法描述中,方括号([和])之中的项为可选择命令参数,大写字母为参数标记,小写字母为参数。可选择命令参数的顺序是任意的。当命令所选用的有关驱动器所在槽号、设备号省略时,总是指定最后使用操作的磁盘设备为当前 DOS 命令所执行的设备。

命令参数说明:

1. f　文件名,以字母打头的字符串组成(其中不能含有逗号),文件名的有效长度最多可达 30 个 ASCII 字符。

2. Ss　设定软盘驱动器所在的槽口号,s 值可以是 1 ~ 7 之间的整数(但不允许为 3)。

中华学习机盘驱接口上所接盘驱的槽口号 s=6。

3. Dd　设定软盘驱动器在盘驱接口卡上的设备号。d 值必须是 1 或 2。

CEC —2000 机最多可接二台盘驱。驱动器 1

口上所接盘驱的设备号 d=1。

4. Vv 软盘片的卷号，一张软盘片只能有一个卷号，卷号的值 v 可以是 1—254 之间的整数。

卷号是在用 INIT 命令进行盘片初始化操作时就已确定了的，是不可更改的，卷号起着核对盘片的作用。

5. Bb 用以指明文本文件起始字符的位置，b 值可以是 0～32767 之间的整数。

6. Rr 在顺序文件中用来指明数据域起始标志，而在随机文件中则用来指明记录数。r 值可以是 0～32767 之间的整数。

3.2 DOS命令调用方法

在没有引导 DOS 操作系统情况下，只能使用 ROM 中的 CEC-BASIC 解释程序。而在引导 DOS 操作系统后，便可在 BASIC 方式下来调用 DOS 命令。其调用方式有二种：1.立即方式，2.程序方式。

3.2.1 立即执行方式

在 BASIC 提示符下，直接打入 DOS 命令，DOS 将立即解释执行该命令。这种工作方式类似于 BASIC 中的立即型命令。

可以立即执行的 DOS 命令有：

BLOAD	BRUN	BSAVE	CATALOG	CLOSE
DELETE	EXEC	FP	INIT	IN #
INT	LOAD	LOCK	MAXFILES	MON
NOMON	PR #	RENAME	RUN	SAVE
UNLOCK	VERIFY			

3.2.2 程序调用方式

在 BASIC 程序中,DOS 操作系统命令均可作为一条特殊的 BASIC 语句被调用。它与一般 BASIC 语句不同的是:DOS 操作系统命令必须通过一条特殊的 PRINT 语句来执行。这条 PRINT 语句将 DOS 命令及其命令参数作为一输出行进行输出,而在这输出行的第一个字符是 CTRL-D。

CTRL-D 是 DOS 命令的引导标记,当 BASIC 解释程序执行 PRINT 语句时,遇到第一个输出字符为 CTRL-D,则作为 DOS 命令去处理,它不会将 CTRL-D 字符及其后面的 DOS 命令字符串作为一般的输出字符显示到屏幕上或输出设备上(当然,如果系统中没有启动 DOS 操作系统,BASIC 程序是不会对 CTRL-D 字符作特殊处理的)。

调用 DOS 命令的语句格式:

PRINT "CTRL-D DOS 命令和命令参数"

或 PRINT CHR $ (4);"DOS 命令和令令参数"

对于整数 BASIC 程序,由于没有 CHR $ (4)函数,所以只能用第一种语句格式,对于 APPLESOFT 程序二种语句格式均可以使用。

如果我们把 CTRL-D 字符先赋值给串变量,上述语句可改写为:

D $ ="CTRL-D"或 D $ =CHR $ (4)

PRINT D $;"DOS 命令和命令参数"

下面这段程序将显示盘片上的文件目录:

```
10  REM DOS 3.3 COMMAND
20  PRINT "CATALOG OF DOS 3.3 SYSTEM
    MASTER"
30  PRINT CHR $ (4);"CATALOG"
```

```
40  END
```

在使用 BASIC 语句调用 DOS 命令时应注意:

(1) 在一个 PRINT 语句输出行中,只能含有一条 DOS 命令,而在这个输出行中必须以 CTRL-D 字符打头,以回车字符(CTRL-M)表示 DOS 命令的结束.

请比较一下下面二段程序为什么不执行 DOS 命令和出现语法错的原因:

例一:
```
10  REM DOS 3.3 COMMAND
20  PRINT "CATALOG OF DOS 3.3
    SYSTEM MASTER";
30  PRINT CHR$(4);"CATALOG"
40  END
```

例二:
```
10  REM DOS 3.3 COMMAND
20  PRINT CHR$(4);"CATALOG",
30  PRINT CHR$(4);"CATALOG"
40  END
```

仅能以程序调用方式执行的 DOS 命令有:

APPEND　　CLOSE　　OPEN　　POSITION

READ　　WRITE

3.3 初始化命令

INIT 命令为初始化命令。

用途:初始化软盘片。

格式:INIT f〔,Ss〕〔,Dd〕〔,Vv〕

说明:该命令将对软盘片进行初始化,并把现已建立在内存中的 BASIC 程序作为 DOS 操作系统引导后,装入并运行

的第一个程序，存入软盘，冠以文件名 f。

INIT 命令一般仅用于对初次使用的空白盘片进行初始化，但当某张已使用过的软盘片上所存储的文件已全部无用，并打算将该盘片用来重新存放一些新文件时，亦可对其进行初始化操作。执行初始化操作后，原盘片上的所有信息都将被清除。

INIT 命令将对软盘执行以下操作：

(1) 对盘片格式化，删除盘片上所有信息，在磁盘表面划分磁道和扇区段。并将所有区段内容写成 0。

(2) 将 DOS 操作系统存放到软盘的 $ 00 ~ $ 02 三个磁道上。

所有经过初始化的软盘上均装有 DOS 操作系统，也均可以作为 DOS 系统的引导盘。

(3) 在 $ 11 道上为该盘片建立文件目录区，填写该盘所属卷号。

卷号可以通过 Vv 参数由用户自行指定，当 Vv 参数省略，或 v=0 时，将自动分配一个卷号为 254，一张盘片只能有一个卷号，一旦建立就不能被修改。

(4) 将当前已建立在主机内存中的 BASIC 程序作为引导 DOS 操作系统后，自动装入执行的第一个程序(通常称为问候程序)存放到软盘上，冠以指定的文件名 f。

(5) 用 INIT 命令初始化后的软盘，只能是“从属盘”，它不能像 DOS 3.3 系统主盘一样可以在 16 ~ 64KB RAM 的任一类型的 APPLE Ⅱ 机上进行启动，但可以通过 MASTER CREATE 程序的运行使其变为系统主盘(详见第四章第 8 节)。

例如，下面这些步骤的操作将初始化一张空白盘片：

(1) 用 DOS 3.3 系统主盘进行启动。

(2) 清除内存,并打入如下程序:

```
10   REM HELLO PROGRAM
20   PRINT "THIS HELLO PROGRAM CREATED
     BY"
30   PRINT "BILL JONES         11/25/86"
40   END
```

(3) 打入命令 RUN,检查运行结果:

```
]RUN
THIS HELLO PROGRAM CREATED BY
BILL JONES                   11/25/86
```

(4) 取出系统主盘,插入空白盘片。

(5) 打入初始化命令:

```
]INIT HELLO,V10
```

(6) 从盘驱中取出软盘片,贴上临时标签,不用时,及时插入磁盘的纸套中,并立放在盘片盒中。

3.4 管理磁盘文件的 DOS 命令

3.4.1 CATALOG

用途:列磁盘文件目录。

格式:CATALOG [,Ss] [,Dd] [,Vv]

说明:该命令将按命令所设定的软盘驱动器列出该盘驱上盘片的文件目录。目录中包括:磁盘的卷号,文件的名称,文件所占用盘片的扇区数,文件的类型和文件的封锁标志等等。例如:

```
]CATALOG
```

DISK VOLUME 254 ———— 磁盘卷号

* A 008 HELLO

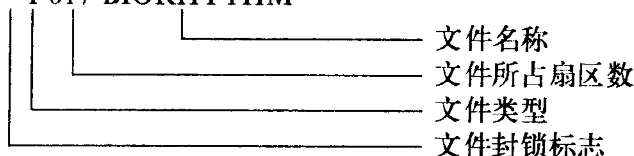
* I 018 ANIMALS

T 003 APPLE PROMS

* I 006 APPLESOFT

* I 026 APPLEVISION

* I 017 BIORHYTHM



3.4.2 LOCK

用途: 置文件封锁。

格式: LOCK f〔,Ss〕〔,Dd〕〔,Vv〕

说明: 该命令将对指定的文件置封锁(写保护)状态,使得该文件不会因意外的写操作而遭破坏。

对某一文件执行封锁操作后,列磁盘文件目录,该文件类型标志的左边将有一星号标志(*),对于被封锁的文件,用户只能执行 LOAD / READ / EXEC / VERIFY 操作,而不能对其执行 SAVE / WRITE / RENAME / DELETE 等操作。否则将会得到信息: FILE LOCKED。

例如,插入刚刚初始化完毕的工作盘,列出磁盘文件目录

〕CATALOG

DISK VOLUME 010

A 002 HELLO

现在执行 LOCK HELLO 命令,再次列出磁盘文件目录时,HELLO 文件将被封锁。

```
]LOCK HELLO
]CATALOG
DISK VOLUME 010
 * A 002 HELLO
```

3.4.3 UNLOCK

用途:解除文件封锁。

格式:UNLOCK f〔,Ss〕〔,Dd〕〔,Vv〕

说明:该命令将解除对文件的封锁,这样可以使得对软盘上的该文件进行删除、重新命名、修改或重写等操作,解除文件封锁后,该文件的封锁标志将消失。

LOCK 和 UNLOCK 命令可以对任意类型的文件进行操作。

3.4.4 RENAME

用途:修改文件名。

格式:RENAME f1,f2 〔,Ss〕〔,Dd〕〔,Vv〕

说明:该命令将对指定的文件重新命名。用文件名 f2 取代原有的文件名 f1,改名后的文件其内容不受影响,可以对任意类型文件执行该操作。

这里的 f1 是一个已在软盘上存在的文件名,f2 必须是软盘上所没有的新文件名,f1 文件必须不被封锁。

例:若工作盘上的文件目录为:

```
]CATALOG
DISK VOLUME 010
* A 002 HELLO
  A 009 ABC
```

现欲将 ABC 文件改名为 BCD,可执行如下命令:

```
]RENAME ABC,BCD
```

]CATALOG

DISK VOLUME 010

* A 002 HELLO

A 009 BCD

注意:RENAME 命令并不检查文件名 f2 是否在软盘上已存在,因此有可能通过 RENAME 命令而使得在一张软盘上有多个同名文件,最好要避免这种不必要的混乱出现。

3.4.5 VERIFY

用途:文件校验。

格式:VERIFY f〔,Ss〕〔,Dd〕〔,Vv〕

说明:该命令将通过读操作对指定文件 f 的校验和进行检查,以确定该文件在软盘上的信息是否正确。

我们知道,DOS 操作系统在把信息存到软盘上的过程中,首先把信息存入文件缓冲区,待存满 256 个字节时,DOS 再把这些信息作为一个数据块存到软盘的扇区中。这时,存到该扇区中的信息除了 256 个字节是文件信息外,还有一个作为"检查和"的校验字节,根据这个校验字节,可以检查文件传输的正确性。

一般说来,从主机写到软盘上的信息都是很可靠的,通过 DOS 命令的使用是可以重新将它读回到内存中来。但是由于人为因素,盘片长期存放,软盘片部分地遭到损伤,使得信息被丢失,DOS 不得将它读回到内存中来。通过 VERIFY 命令可以对盘上的文件进行一次读操作,校验一下该盘片上文件的正确性。

VERIFY 命令操作过程中,如果被校验的文件正确,将得不到任何回答信息,否则,将给出 I/O ERROR 信息。这将提示你,该文件已遭破坏,这时应考虑重新复制该文件,或

者将该盘上的其它文件盘制到其它软盘上，以避免整张盘片被损坏。DOS 3.3 操作系统在存储每个文件时，都将自动执行一次 VERIFY 操作，以保证存盘操作是成功的。

VERIFY 命令可适用于对任何类型的文件进行校验操作，该操作过程将不破坏主机内存中程序和数据，也不影响软盘上的文件。

3.4.6 DELETE

用途：删除文件。

格式：DELETE f〔,Ss〕〔,Dd〕〔,Vv〕

说明：该命令将从软盘上删除所指定的文件，这个文件可以是任意类型的，并且文件是不被封锁的。

如果要删除的文件不存在，将出现信息：FILE NOT FOUND。如果该文件被封锁，将得到信息：FILE LOCKED。

3.5 与 BASIC 程序有关的 DOS 命令

3.5.1 LOAD

用途：装入 BASIC 程序。

格式：LOAD f〔,Ss〕〔,Dd〕〔,Vv〕

说明：该命令将从软盘上装入一个类型为 A 或 I 的 BASIC 程序。

如果命令中所指出的文件名不在软盘上，将得到信息：FILE NOT FOUND。如果欲装入的文件，其文件类型不是 A 或 I，则会得到信息：FILE TYPE MISMATCH。如果没有用 DOS 3.3 系统主盘进行启动，或没有将整数 BASIC 解释程序装入到主机内存中，而要装入的却是 I 类型文件，将会得

到信息：LANGUAGE NOT AVAILABEL。

LOAD 命令装入 BASIC 程序之前，首先要检查磁盘文件的类型，根据装入的是 A 类型文件还是 I 类型文件，转入 APPLESOFT 解释程序或整数 BASIC 解释程序，然后才将该文件的内容装入到主机内存中。

在装入 BASIC 程序时，盘上的文件保持不变，原先已在内存中的 BASIC 程序将被清除。

3.5.2 SAVE

用途：保存 BASIC 程序。

格式：SAVE f〔,Ss〕〔,Dd〕〔,Vv〕

说明：该命令将把当前已建立在主机内存中的 BASIC 程序保存到软盘上，以建立程序副本。

f 指出的是程序副本的文件名。文件类型根据当前系统处于何种 BASIC 语言状态所确定。如果该盘上没有该文件名，则在软盘上创建该文件。如果该文件名在软盘上已存在，并且文件类型与系统的当前状态相吻合，则用当前已在内存中的 BASIC 程序替代原有文件。如果软盘上的文件类型和主存中 BASIC 程序类型不一致，则给出信息：FILE TYPE MISMATCH。

利用 SAVE 命令，我们可以在一个软盘上建立多个程序副本，也可以在多个软盘上建立多个程序副本。

例如，我们把在 DOS 3.3 系统主盘上的 COLOR DEMO 程序存到工作盘上，可以执行如下操作：

(1) 用 DOS 3.3 系统主盘进行启动。

(2) 打入命令：LOAD COLOR DEMO

(3) 换上工作盘，打入命令：SAVE COLOR DEMO

这样，在你的工作盘上，便保存了一个和 DOS 3.3 系统

主盘上一模一样的整数 BASIC 程序:COLOR DEMO。

3.5.3 RUN

用途:运行软盘上的 BASIC 程序。

格式:RUN f〔,Ss〕〔,Dd〕〔,Vv〕

说明:该命令将把软盘上的某一 BASIC 程序 f 装入到内存并运行之。

该命令的执行步骤是:①执行 DOS 的 LOAD 命令,②执行 BASIC 的 RUN 命令。

3.6 语言之间进行转换的DOS命令

DOS 3.3 操作系统为方便 APPLE Ⅱ机不同版本用户的使用提供了 APPLESOFT 解释程序和整数 BASIC 解释程序二种语言状态相互切换的命令。

3.6.1 INT

用途:进入整数 BASIC。

格式:INT

说明:该命令使系统进入整数 BASIC 状态,并清除内存中现有程序,恢复有关程序指针。

如果我们想从 APPLESOFT 状态进入整数 BASIC 状态,可以使用这一命令。该命令不带任何参数,它不能设定软盘驱动器的槽号和设备号。

如果机内没有装入整数 BASIC 解释程序,系统将给出信息:LANGUAGE NOT ANAILABEL。

3.6.2 FP

用途:进入 APPLESOFT。

格式:FP〔,Ss〕〔,Dd〕

说明：APPLESOFT 程序又称为浮点 BASIC 程序，该命令将使得系统进入 APPLESOFT 状态。并清除内存中现有的 BASIC 程序，恢复各有关指针。

如果我们在运行了某一整数 BASIC 程序，想回到 APPLESOFT 状态可以打入 FP 命令，如果我们想清除当前内存中的 BASIC 程序，可以打入 FP 命令。如果我们因运行了某个 BASIC 程序，使得系统有关指针的改变，现在想装入第二个程序执行时，系统回答"OUT OF MEMORY"信息拒绝执行时，可以使用 FP 命令。

3.7 对文本文件进行操作的 DOS 命令

文本文件是用来存放数据信息的文件，其数据信息可以是：运算数据、计算的结果、信件的副本、帐单和表格等，也可以是组成 DOS 命令和 BASIC 命令的字符串。

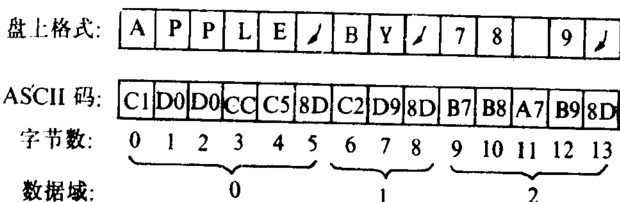
文本文件是通过程序方式调用 DOS 命令来创建和检索的，它可以通过某一程序来创建，而由另一程序进行检索。

文本文件按其存取方式可以分成顺序文件和随机文件。这两种文件在目录中的类型标识符均为 T，但在使用方法和存储格式上却有所不同，如果使用不当，可使得存取过程中所得到的数据结果与预期的大不相同，甚至会得到出错信息。

3.7.1 顺序文件

顺序文件由数据域组成，每个数据域由回车(CR)字符作为终结标志，它通常由不带逗号和分号的 PRINT 语句自动生成，也可以由输出 CHR $ (13)字符来产生。顺序文件在软盘上是连续存放的，各个数据域之间没有间隙。其数据域的长度和文件长度可以是任意的。

顺序文件在盘上的存放格式如图所示:



一、顺序文件的一个例子

假设我们想把单词: APPLE、BANANA、FRUIT、EAGLE 和 GOOSE 作为一个文本文件保存到软盘上,我们可以用如下的程序来完成:

```
10  REM MAKE-WORD1
15  DIM A $ (10)
20  D $ = CHR $ (4): REM CTRL-D
30  PRINT D $ ;"OPEN WORD1"
40  PRINT D $ ;"WRITE WORD1"
50  FOR I = 1 TO 5: READ A $ (I): PRINT
    A $ (I): NEXT I
60  PRINT D $ "CLOSE WORD1"
70  END
100 DATA "APPLE","BANANA","EAGLE"
    ,"GOOSE","FRUIT"
```

在这个程序中,我们使用了三条 DOS 命令: OPEN、WRITE 和 CLOSE,其作用是: OPEN 创建一文本文件 WORD1,并把该文件建立在目录中,WRITE 命令对 WORD1 文件执行写操作,使得第 50 列语句中的 PRINT 命

令将把输出信息送到文件 WORD1 中，而不是屏幕上，CLOSE 命令则是停止对 WORD1 文件执行写操作。

如果我们不是在 DOS 的监视命令方式下运行该程序，屏幕上将看不到任何信息，如果我们打入置监视方式命令：MON C,I,O，然后再打入 RUN 命令，则会在屏幕上看到如下信息：

```
]MON C,I,O
]RUN
OPEN WORD1
WRITE WORD1
APPLE
BANANA
EAGLE
GOOSE
FRUIT
CLOSE WORD1
```

这样我们可以确信已把这些单词保存到了 WORD1 文件中，那么如何再从这个文件中读回这些信息呢？下面这个程序将帮助你解决这个问题：

```
]LIST
10  REM RETRIEVE-WORD1
15  DIM B$(10)
20  D$ =CHR$(4): REM CTRL-D
30  PRINT D$;"OPEN WORD1"
40  PRINT D$;"READ WORD1"
50  FOR I =1 TO 5: INPUT B$(I): NEXT I
```

```
60  PRINT D$"CLOSE WORD1"
70  END
```

在这个程序中，我们同样也使用了三条 DOS 命令：OPEN、READ 和 CLOSE。这里的 OPEN 命令将打开 WORD1 文件，READ 命令将告诉 DOS：第 50 行语句中的 INPUT 命令将从 WORD1 文件中读取数据，而不是从键盘上。CLOSE 命令将关闭 WORD1 文件，结束对 WORD1 文件执行读操作。

如果打入 MON C,I,O 命令，再运行该程序，我们可以看到如下信息：

```
]MON C,I,O
]RUN
OPEN WORD1
READ WORD1
?APPLE
?BANANA
?EAGLE
?GOOSE
?FRUIT
CLOSE WORD1
```

为了进一步确认 WORD1 文件中的数据被读了回来，我们可打入如下命令：

```
]PRINT B$(1),B$(3),B$(5)
APPLE       EAGLE       FRUIT
```

通过这二个例子我们看到了顺序文件的写过程和读过程 与它所使用的 DOS 命令有着密切的关系。

二、OPEN

用途:打开顺序文件。

格式:OPEN f〔,Ss〕〔,Dd〕〔,Vv〕

说明:该命令将打开一顺序文件f,如果该文件不存在,则在磁盘的目录中创建该文件。

在对某一文件进行读操作或写操作之前,首先必须打开该文件。此时DOS将给该文件分配595个字节的文件缓冲区,供磁盘与主存之间的信息交换用。程序一次允许同时打开的文件个数是由MAXFILES命令所决定,最多不得超过16个;DOS启动后将自动执行MAXFILES3命令,此时系统允许打开的文件个数最多为3个。

三、CLOSE

用途:关闭文件。

格式:CLOSE〔f〕

说明:对某一文本文件在完成读/写操作后必须及时关闭该文件,使得DOS命令收回分配给该文件的文件缓冲区,使它可以用于新打开的文件。否则会由于打开的文件太多,而出现:NO BUFFER AVAILABLE信息。

该命令执行时,f指出了要关闭的文件名,当文件名省略时,将关闭所有已打开的文件。

四、WRITE

用途:写顺序文件。

格式:WRITE f〔,Bb〕

说明:该命令将对已打开的文件f执行写操作,参数Bb指明从顺序文件的第b个字节开始写数据,如果参数Bb省略则从顺序文件的第0个字节开始写数据。执行该命令后,所有PRINT语句输出的信息都将存到文件中,而不是显示到屏幕上。

注意:如果打开了盘上已有的文本文件,再对它进行执行写操作,此时除非新写入的数据多于原有文件的信息,并覆盖了原有的信息,否则得到的文件内容将是新写入的信息和原来文件混合构成。

在下面这个例子中,我们虽然在 WORD1 文件中只存储了二个数据,但因原来已有 5 个数据,所以从该文件中读到了 5 个数据。

```
10  REM MAKE WORD1
15  DIM A $ (10), B $ (10)
20  D $  = CHR $ (4): REM CTRL-D
30  PRINT D $ ; "OPEN WORD1"
40  PRINT D $ ; "WRITE WORD1"
50  FOR I = 1 TO 2:  READ A $ (I):  PRINT
    A $ (I):  NEXT I
60  PRINT D $ "CLOSE WORD1"
80  DATA 123, 456
100 REM RETRIVE WORD1
110 PRINT D $ ; "OPEN WORD1"
120 PRINT D $ ; "READ WORD1"
130 FOR I = 1 TO 5:  INPUT B $ (I):  NEXT I
140 PRINT D $ "CLOSE WORD1"
150 END
```

打入 MON C,I,O 命令后,打 RUN 命令所以从屏幕上得到如下信息:

```
]MON C,I,O
]RUN
OPEN WORD1
```

```
WRITE WORD1
123
456
CLOSE WORD1
OPEN WORD1
READ WORD1
?123
?456
?NANA
?EAGLE
?GOOSE
CLOSE WORD1
```

通过这个例子，我们知道 WORD1 文件的内容是由二个程序的写入数据混合构成的，为了避免出现这种混乱；使得到的新文件内容只有新写入的数据，可以采用打开—删除—打开的方式来建立新文件。在该例中的 30 行语句前面增加二条语句：

```
22  PRINT D $ ;"OPEN WORD1"
24  PRINT D $ ;"DELETE WORD1"
```

这样无论磁盘上是否有 WORD1 文件，都不会出现混乱情况。

五、READ

用途：读顺序文件。

格式：READ f 〔, Bb〕

说明：该命令将对已打开的文件 f 执行读操作，参数 Bb 指明了从顺序文件的第 b 个字节开始读数据，如果参数 Bb 省略则从顺序文件的第 0 个字节开始读数据。执行该命

令后，所有 INPUT 语句或 GET 语句都将从文件中读到数据，而不是从键盘上读数据。

注意：在用 INPUT 语句读入数据时，一次读入一个域，并把这个域中以逗号分隔的数据串依次赋给 INPUT 语句中的各个变量。

如果变量的类型和所给的数据类型不匹配，则给出 ?REENTER 信息，继续取下一个数据，如果数据域中的数据多于 INPUT 语句中的变量个数，则给出信息：?EXTRA IGNORED，忽略多余的数据部分。如果数据域中的数据个数少于 INPUT 语句中变量的个数时，屏幕将出现??信息，自动到下一个数据域中取数据。这种情况的出现和 BASIC 语句一样，是我们所不希望的错误，会使得读数据丢失。

防止读数据出错的有效办法是：检索程序读数据的格式应按照写数据文件的格式来设计，变量类型也应和写数据程序的变量类型相一致。

六、APPEND

用途：使文件添加内容。

格式：APPEND f 〔,Ss〕〔,Dd〕〔,Vv〕

说明：该命令将自动打开顺序文件 f，并把文件位置指针移到文件的末尾(即文件最后一个字符的后面)。很明显，使用该命令的主要目的是在文件的末尾增加新的信息。

在 APPEND 命令之后必须使用 WRITE 命令，如果使用了 READ 命令，执行 INPUT 语句时，则会显示信息：END OF DATA。

下面这个例子将在 WORD1 文件中增加 2 个单词，SHEEP 和 WOOD。

```
10  REM APPEND-WODR1
```

```
20  D$ = CHR$ (4)
30  PRINT D$;"APPEND WORD1"
40  PRINT D$;"WRITE WORD1"
50  PRINT "SHEEP"
60  PRINT "WOOD"
70  PRINT D$;"CLOSE WORD1"
80  END
```

相应的检索程序可改为:

```
10  REM RETRIEVE-WORD1
15  DIM B$ (10)
20  D$ = CHR$ (4)
30  PRINT D$;"OPEN WORD1"
40  PRINT D$;"READ WORD1"
50  FOR I = 1 TO 7: INPUT B$ (I): NEXT I
70  PRINT D$;"CLOSE WORD1"
80  END
```

七、POSITION

用途:文件的定位。

格式:POSITION f 〔,Rr〕

说明:该命令用来对顺序文件执行定位操作,它可以把文件位置指针从当前位置移到下面第 r 个域的起始位置。参数 Rr 指明的是当前位置的第 r 个域。当参数 Rr 省略时,表示不移动文件的位置指针。

POSITION 命令必须作用于一个已打开的顺序文件,文件名 f 必须与 OPEN 命令中的文件名一致,它和其它 DOS 命令一样,将撤消 WRITE 和 READ 命令的功能,所以,POSITION 命令必须用在 WRITE 和 READ 命令之前,

否则，使用过 POSITION 命令之后还需要重新使用 READ 或 WRITE 命令，才能执行读操作或写操作。

例：在 MAKE-WORD1 例子程序中，我们已建立了一个顺序文件 WORD1，它由五个域构成，每一个域包含了一个单词。APPLE 单词在文件中是第 0 域。现在我们想从 WORD1 文件中读取第三、四两个域中的单词，可用下面程序来实现。

```
10  REM POSITION-WORD1
20  D$ = CHR$(4)
30  PRINT D$;"OPEN WORD1"
40  PRINT D$;"POSITION WORD1,R3"
50  PRINT D$;"READ WORD1"
60  INPUT A$
65  INPUT B$
70  PRINT D$;"CLOSE WORD1"
80  PRINT A$: PRINT B$
90  END
```

打入 RUN 命令可得到：

```
]RUN
GOOSE
FRUIT
```

3.7.2 随机文件

随机文件由记录组成，它以记录为单位进行数据的读操作和写操作。每个记录中可以包含有许多数据域，各个记录的记录长度是固定的，一个记录的最大长度是 32767 个字符。

设有一随机文件，它由三个记录构成，记录长度为 6，它在盘上的存放格式为：

盘上格式:	A	P	P	L	E	↙	B	Y	↙				7	↙	8	↙	9	↙
ASCII 码:																		
文件字节:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
记录字节:	0	1	2	3	4	5	0	1	2	3	4	5	0	1	2	3	4	5
数据域:	0						0						0		1		2	
记录:	0						1						2					

从这里可以看出顺序文件是随机文件的一个特例，其中所含的记录个数为 1。

使用随机文件，可以实现快速地查找、增补或修改其中任一记录中的数据，而不影响其它记录中的内容。因此，象帐目管理一类的应用软件特别适用于随机文件。

一、OPEN

用途：打开随机文件。

格式：OPEN f, Lj 〔,Ss〕〔,Dd〕〔,Vv〕

说明：该命令将打开一随机文件 f，其记录长度由参数 Lj 来决定，j 是 1 — 32767 之间的数。如果文件 f 不存在，则在磁盘目录中建立该文件。

在打开随机文件时，DOS 并不把该文件的记录长度存入磁盘，如果我们重新打开 f 文件时，参数 Lj 所指明的长度与创建时指明的长度不一致，则 DOS 将按照新给的参数来计算文件中各记录的位置，这样读写随机文件时可能会造成混乱的结果。为了避免出现这种情况，我们必须保存好关于随机文件的结构和数据格式，以便在编制检索程序时不至于搞

错。最为简单的办法是把这些信息作为文件的第0个记录保存起来，而把记录长度附在文件名中。例如：MAILER文件可取名为MAILER.L30或MAILER(L30)等。

二、CLOSE

该命令的用途和使用格式与3.7.1节CLOSE完全相同。

三、WRITE

用途：写随机文件。

格式：WRITE f〔,Rr〕〔,Bb〕

说明：该命令将对随机文件f执行写操作。这里有二个参数Rr和Bb，分别用来指明是对随机文件中第r个记录执行写操作，其起始位置为b字节。

参数Rr指定文件的r个记录，r值可以是0～32767之间的一个数，当r=0或省略该参数时表示对f文件的第0个记录执行写操作。

参数Bb表示从该记录中第b个字节开始执行写操作。如果b值很大，超出了记录长度，则在使用PRINT语句后，写入的数据可能会破坏下一个记录中的内容。

在使用WRITE命令后，不能使用其它DOS命令或INPUT语句或GET语句，否则将会终止WRITE命令的执行。

四、READ

用途：读随机文件。

格式：READ f〔,Rr〕〔,Bb〕

说明：该命令将对随机文件f执行读操作。这里有二个参数Rr和Bb，分别用来指明是对随机文件中第r个记录执行读操作，其起始位置为b字节。

在使用 READ 命令后，不能使用其它 DOS 命令或 PRINT 语句，否则将会终止 READ 命令的执行。

五、随机文件的一个例子

这里给出一个创建和使用随机文件的例子。在这个例子中，随机文件的记录长度为 30 个字符，每个记录中包含有 3 个域，即姓名 N $，电话号码 P $ 和邮政编码 Z $。

```
10  REM MAKE MAILER
15  DIM N$(10),P$(10),Z$(10)
20  D$ =CHR$(4)
22  PRINT D$;"OPEN MAILER,L30"
25  PRINT D$;"DELETE MAILER"
30  PRINT D$;"OPEN MAILER,L30"
40  FOR I =1 TO 5
50  PRINT D$;"WRITE MAILER,R";I
60  READ N$(I),P$(I),Z$(I)
70  PRINT N$(I): PRINT P$(I): PRINT
    Z$(I)
80  NEXT I
90  PRINT D$;"CLOSE MAILER"
100 DATA ZHANG,28-2451,100010
102 DATA WANG,66-8841,100012
104 DATA ZHOU,5-2300,210031
106 DATA ZEN,1-7001,100015
107 DATA XIU,5-4321,210033
110 END
```

在内存中建立完该程序后，打入 RUN 命令，即在 MAILER 文件中建立含有 5 个记录的随机文件。

下面这段程序，将把 MAILE 文件的记录取出来，并显示到屏幕上。

```
10  REM RETRIEVE MAILER
15  DIM N $ (10), P $ (10), Z $ (10)
20  D $  = CHR $ (4)
30  PRINT D $ ; "OPEN MAILER, L30
40  FOR I = 1 TO 5
50  PRINT D $ ; "READ MAILER, R"; I
60  INPUT N $ (I): INPUT P $ (I):  INPUT
    Z $ (I)
70  NEXT I
80  PRINT D $ ; "CLOSE MAILER'
90  FOR I = 1 TO 5:  PRINT N $ (I), P $
    (I), Z $ (I):  NEXT I
100 END
```

打入 RUN 命令后，我们将得到如下信息：

```
]RUN
ZHANG        28-2451        100010
WANG         66-8841        100012
ZHOU         5-2300         210031
ZEN          1-7001         100015
XIU          5-4321         210033
```

3.7.3 EXEC

用途：执行顺序文件。

格式：EXEC f 〔, Rr〕〔, Ss〕〔, Dd〕〔, Vv〕

说明：该命令将从顺序文件 f 中第 r 个域开始执行该文件中所含的命令。这些命令就象是从键盘上一句一句打入的一

样。

在这个命令中，参数 Rr 指出文件 f 的一个绝对域址，如果参数 Rr 省略则表示从文件的开头去执行。

顺序文件 f 中所含的命令可以是 BASIC 命令、监控命令和 DOS 命令。下面这个例子中将建立一个含有 DOS 命令、BASIC 命令和监控命令的文本文件 TEST。

```
10   REM MAKE-EXEC
20   D $  = CHR $ (4)
30   PRINT D $ ;"OPEN TEST"
40   PRINT D $ ;"WRITE TEST"
50   PRINT "CATALOG"
60   PRINT "CALL-151"
70   PRINT "FD18L"
80   PRINT "FP"
90   PRINT "?34+ 16"
100 PRINT D $ ;"CLOSE TEST"
110  END
```

打入这段程序后，打入命令 RUN。此时，在软盘上建立了可执行的顺序文件 TEST。

打入命令 EXEC TEST，屏幕上将出现信息：

```
]EXEC TEST
]
DISK VOLUME 254
A 002 HELLO
A 002 MAKE-WORD1
T 002 WORD1
A 002 RE-WORD1
```

```
A 003 MAK-RE
A 002 MAKE-EXEC
T 002 TEST
]
 *
FD18-   6C   38   00   JMP    ($ 0038)
FD1B-   E6   4E        INC    $ 4E
FD1D-   D0   02        BNE    $ FD21
FD1F-   E6   4F        INC    $ 4F
FD21-   2C   00   C0   BIT    $ C000
FD24-   10   F5        BPL    $ FD1B
FD26-   91   28        STA    ($ 28),Y
FD28-   AD   00   C0   LDA    $ C000
FD2B-   2C   10   C0   BIT    $ C010
FD2E-   60             RTS
FD2F-   20   0C   FD   JSR    $ FD0C
FD32-   20   A5   FB   JSR    $ FBA5
FD35-   20   0C   FD   JSR    $ FD0C
FD38-   C9   9B        CMP    # $ 9B
FD3A-   F0   F3        BEQ    $ FD2F
FD3C-   60             RTS
FD3D-   A5   32        LDA    $ 32
FD3F-   48             PHA
FD40-   A9   FF        LDA    # $ FF
FD42-   85   32        STA    $ 32
 *
]
```

50

}

在使用 EXEC 命令时要注意以下几点:

(1) 如果在 f 文件中含有 EXEC 命令,假设为"EXEC g",那么当执行到 EXEC g 命令时,将自动关闭 f 文件,而打开并执行 g 文件。

(2) 如果在 f 文件中含有 DOS 的 RUN 命令,假设为"RUN g",那么当执行到 RUN g 命令时,系统将执行 RUN g 操作,等 g 程序执行完毕后,再继续执行 f 文件中的后续操作命令。但要注意,在 EXEC 命令下运行程序 g 时,程序中的任何 INPUT 语句都会将文件 f 的下一个域作为输入数据,而不从键盘上输入数据。

(3) 如果在文件 f 中含有带行号的 BASIC 程序行,EXEC 命令并不执行这些程序行中的命令,而是按其行号次序将它们装入内存中,就象这些 BASIC 语句行是从键盘上输入到内存中一样。

利用 EXEC 的这一特性,我们可以把某一子程序插入到主程序中,修改部分程序,或把几个程序合并为一个程序。也可以把整数 BASIC 程序转换成 APPLESOFT 等等。

关于使用 EXEC 命令来执行 DOS 命令和 BASIC 语句,DOS 3.3 系统主盘上给出了一个 BASIC 程序:"EXEC DEMO"作为范例,有兴趣的读者可以分析一下该程序所做的工作。

3.8 对 2 进制文件进行操作的 DOS 命令

DOS 操作系统除了可以对 BASIC 程序文件、文本文件

进行操作处理外,还能够对2进制文件进行操作。在文件目录中,2进制文件的类型标志是B文件。

2进制文件存放的是主机内存中的2进制信息副本,即把主机内存中某个地址段的信息以文件的形式保存在软盘上。这些文件信息可以是机器语言程序,也可以是2进制数据或者是图象信息等。2进制文件在软盘上的存储格式为:

地址　　长度　　　2进制数据块

DOS操作系统对2进制文件提供了三条操作命令。即:BSAVE、BLOAD和BRUN。

3.8.1 BSAVE

用途:保存2进制文件。

格式:BSAVE f,Aa,Lj〔,Ss〕〔,Dd〕〔,Vv〕

说明:该命令将把内存中指定的一段2进制数据,以f为文件名,保存到指定的软盘上。

命令参数Aa指明了要存储的2进制数据在内存中的起始地址。a的值可以在0～65535之间。

命令参数Lj指明了该2进制数据块的字节长度。j的值可以在0～32767之间。

值得注意的是:在此命令中命令参数Aa和Lj均是必须的。参数a和j可以用10进制来表示,也可以用16进制来表示。用16进制表示时,在数值前面要冠以字符“$”。

例:用下面二个命令都可以创建一个名为“PICTURE”的2进制文件,该文件将包含在主机内存中高分辨图象第二页内的图象信息。

```
BSAVE PICTURE, A$4000, L$2000
BSAVE PICTURE, A16384, L8192
```

3.8.2 BLOAD

用途:装入2进制文件。

格式:BLOAD f〔,Aa〕〔,Ss〕〔,Dd〕〔,Vv〕

说明:该命令的作用是将盘上的2进制文件f装入到主机内存的指定区域中。

执行该命令时,系统工作状态不会发生变化,也不会破坏内存中的BASIC程序,除非被装入的2进制文件侵入到BASIC程序所占用的存储空间。

命令参数Aa指明了2进制文件装入到内存区域的起始地址。a的值可以在0~65535之间(16进制则为$0~$FFFF)。当该命令参数省略时,其起始地址就是该文件由"BSAVE"命令创建时所指明的起始地址。

例如,若要将PICTURE文件装回到原来的存储区中(即高分辨图象区的第二页)中,可用命令:

```
BLOAD PICTURE
BLOAD PICTURE,A$4000
BLOAD PICTURE,A16384
```

使用下列命令则将该图象信息装入到高分辨图象区的第一页:

```
BLOAD PICTURE,A$2000
BLOAD PICTURE,A8192
```

要注意,使用"BLOAD"命令将机器语言程序装入内存时,如果指定的存储区与该程序原来建立时的存区不一致时,该程序可能就不能运行了。

3.8.3 BRUN

用途:装入并运行机器语言程序。

格式:BRUN f〔,Aa〕〔,Ss〕〔,Dd〕〔,Vv〕

说明:该命令首先将装入以 f 为文件名的 2 进制文件,然后转移(JMP)到该文件的第一个字节去执行程序。

在使用该命令时要做到以下几点才不致于引起系统的混乱和"挂起"。

(1) 必须确保 f 是由机器语言程序所组成的 2 进制文件。

(2) 该程序的入口地址是该文件中的第一个字节。

(3) 如果在命令中使用了命令参数 Aa,则参数 a 所给出的地址要和建立该文件时起始地址相一致。

3.9 辅助命令

3.9.1 MON

用途:置监视方式。

格式:MON〔C〕〔,I〕〔,O〕

说明:在用程序方式调用 DOS 命令时,DOS 命令及主机与磁盘之间的信息交换均不在屏幕上显示出来,但在调试程序时,监视这些信息却是十分必须的,通过它我们可以找出程序中的问题。MON 命令可以满足我们监视这些信息的要求。

在该命令中有三个参数可供我们选择,它们分别代表命令信息、输入信息和输出信息:

C 显示所使用的 DOS 命令。

I 显示由磁盘读入主机的文本信息。

O 显示由主机输出到文本文件中的信息。

这三个参数可以按任何顺序和任意组合形式出现,这取决于希望监视的信息。如果这三个参数均不出现,则

MON 命令不起任何作用。

例: 下面这个例子将显示 DOS 命令和从主机写到软盘上的数据

```
10  D$ =CHR$(4):REM CTRL-D
20  DIM A$(10)
30  NAME$ ="ANIMALS"
40  PRINT D$;"MON C,O"
50  PRINT D$;"OPEN";NAME$
60  PRINT D$;"WRITE";NAME$
70  FOR I=1 TO 3:READ A$: PRITN A$:
    NEXT I
80  PRINT D$;"CLOSE";NAME$
90  FOR I=1 TO 3000: NEXT I
100 PRINT D$;"OPEN";NAME$
110 PRINT D$;"READ";NAME$
120 FOR I=1 TO 3: INPUT A$(I): NEXT I
130 PRINT D$;"CLOSE";NAME$
140 DATA HORSE,CAT,DOG
150 END
```

其运行结果如下所示:

```
]RUN

OPEN ANIMALS
WRITE ANIMALS
HORSE
CAT
DOG
```

```
CLOSE ANIMALS
OPEN ANIMALS
READ ANIMALS
CLOSE ANIMALS
```

〕

该命令执行后一直有效,除使用了 RESET 键或使用了 NOMON、FP、INT 等命令,或者在监控方式下执行了 3DOG 或 3D3G 等操作后,才会使得 MON 命令不起作用。

3.9.2 NOMON

用途:解除监视方式。

格式:NOMON 〔C〕〔,I〕〔,O〕

说明:该命令将撤消对其参数所指明的 DOS 操作信息的监视。参数的含义和使用方法和 MON 命令完全相同。

3.9.3 PR #

用途:置输出设备选择。

格式:PR #S

说明:该命令用来选择输出设备,其中参数 S 表示该设备所占用的槽口号,S 值可以是 0 ~ 7 之间的整数。

执行该命令后,系统的输出信息都将送到 S 值所指明的槽口设备上处理。对于中华学习机来说,S=0 表示输出为屏幕显示,S=3 表示进入汉字系统,S=6 表示从软盘上引导 DOS 操作系统。

例如,CEC -2000 中华学习机打印机槽口号为 1。我们要打印数据,可按下列程序执行操作。

```
10  PRINT CHR $ (4);"PR #1"
```

```
20  A$ ="ABCDEFG0123456789"
30  FOR I =1 TO 3: PRINT A$:NEXT
40  PRINT CHR$(4);"PR #0"
50  END
```

3.9.4 IN

用途：置输入设备选择。

格式：IN #S

说明：该命令用来选择输入设备，其中参数 S 表示该设备所占用的槽口号，S 值可以是 0 ~ 7 之间的整数。

执行该命令后，系统的输入信息都将由 S 值所指定的槽口设备所提供，对于中华学习机来说，S =0 表示输入设备为键盘，S =3 表示进入汉字系统，S =6 表示从软盘上引导 DOS 操作系统。

3.9.5 MAXFILES n

用途：置文件缓冲区的大小。

格式：MAXFILES n

说明：该命令用来设置文件缓冲区的个数，参数值 n 可以是 1 ~ 16 之间的整数。

一个文件缓冲区占用 595 个字节的内存空间，每一个被使用的文件均占用一个文件缓冲区，对某个文件执行读操作时，首先从盘上将一个扇区的信息(256 个字节)读入到该文件的缓冲区中，然后再将其中的一部分送给程序。如果对某个文件执行写操作时，也是把信息送到文件缓冲区，待写满 256 个字节时再送到磁盘上。

文件缓冲区的个数决定了程序一次可以打开的文件个数。启动 DOS 时，系统将自动执行"MAXFILES 3"命令，把文件缓冲区的数目定为 3 个。所以它一次最多可以打开

的文件个数只能是3个文件。如果一次打开的文件个数大于MAXFILES命令中指定参数值n时，将给出信息：NO BUFFERS AVAILABLE。

例如，我们在主机内存中打入如下程序。

```
5    DIM A $ (10), B $ (10)
10   D $ =CHR $ (4): N1 $ ="ABC": N2 $
     ="XYZ"
20   PRINT D $ ; "OPEN"; N1 $
25   PRINT D $ ; "DELETE"; N1 $
30   PRINT D $ ; "OPEN"; N1 $
35   PRINT D $ ; "WRITE"; NI $
40   FOR I=1 TO 5: READ A $ (I): PRINT A $ (I);
     NEXT I
50   PRINT D $ ; "OPEN"; N2 $
55   PRINT D $ ; "DELETE"; N2 $
60   PRINT D $ ; "OPEN"; N2 $
65   PRINT D $ ; "WRITE"; N2 $
70   FOR I=1 TO 5: B $ =LEFT $ (A $ (I), 1):
     PRINT B $ : NEXT I
80   PRINT D $ ; "CLOSE"
90   DATA ABC, BCD, CDE, DEF, EFG, GHI
100  END
```

在打入RUN命令前，我们执行MAXFILES1命令。在DOS监视方式下，我们会得到如下信息：

```
]MAXFILES 1
```

```
]RUN
OPENABC
DELETEABC
OPENABC
WRITEABC
ABC
BCD
DEF
EFG
GHI
OPENXYZ

NO BUFFERS AVAILABLE

Break in 50
```

遇到 NO BUFFERSAVAILABLE 错误时，其解决的唯一办法是关闭文件，我们把上面这段程序稍加改造，其程序及运行结果如下：

```
5    DIM A $ (10), B $ (10)
10   D $  =CHR $ (4): N1 $  ="ABC": N2 $
      ="XYZ"
15   ONERR GOTO 110
20   PRINT D $ ; "OPEN"; N1 $
25   PRINT D $ ; "DELETE"; N1 $
30   PRINT D $ ; "OPEN"; N1 $
35   PRITN D $ ; "WRITE"; N1 $
40   FOR I=1 TO 5:  READ A $ (I):
```

```
    PRINT A $ (I): NEXT I
50  PRINT D $ ; "OPEN"; N2 $
55  PRINT D $ ; "DELETE"; N2 $
60  PRINT D $ ; "OPEN"; N2 $
65  PRINT D $ ; "WRITE"; N2 $
70  FOR I=1 TO 5: B $ =LEFT $ (A $ (I), 1):
    PRINT B $ : NEXT I
80  PRINT D $ ; "CLOSE"
90  DATA ABC, BCD, CDE, DEF, EFG, GHI
100 END
110 PRINT D $ ; "CLOSE": RESUME

]MAXFILES 1

]RUN
OPENABC
DELETEABC
OPENABC
WRITEABC
ABC
BCD
CDE
DEF
EFG
OPENXYZ  CLOSE
OPENXYZ
DELETEXYZ
```

```
OPENXYZ
WRITEXYZ
A
B
C
D
E
CLOSE
```

注意：在执行 MAXFILES 命令时，系统将修改 HIMEM：的值，所以放在 HIMEM 之下的整数 BASIC 程序和 APPLESOFT 程序的字符串变量有可能被破坏。故在使用 MAXFILES 命令时建议：对于 APPLESOFT 程序中使用 MAXFILES 命令时，应把它作为程序的第一条语句，放在任何串变量赋值语句和字串数组定义之前。对整数 BASIC 程序，应放在装入 BASIC 程序之前，使用该命令。

3.9.6 程序的链接

一、整数 BASIC 程序的链接

格式：CHAIN f〔,Ss〕〔,Dd〕〔,Vv〕

说明：该命令只适用于整数 BASIC 程序之间的链接，链接操作的目的是装入运行第二个 BASIC 程序时，第一个程序运行的变量结果仍保留在内存中，可以为第二个程序所使用。

例如，我们可以建立如下二个程序，分别取名为程序 A 和程序 B，保存到工作盘上。

程序 A：

```
5   DIM A(10)
10  FOR I=1 TO 5: A(I) =I * 2:  PRINT A(I),:
```

```
      NEXT I  PRINT
20  PRINT "CHAIN B"
30  END
```

程序 B:

```
10  FOR J =1 TO 5: A(J) =A(J)+ J:  PRINT
      A(J),:  NEXT J:  PRINT
20  END
```

打入命令 MON C 和 RUN,可得到如下结果:

```
>MON C
>RUN A
2      4      6      8      10
CHAIN B
3      6      9      12     15
```

二、APPLESOFT 程序的链接

在 DOS3.3 操作系统中没有提供 APPLESOFT 程序之间的链接操作,但在 DOS3.3 系统主盘上提供了 CHAIN 程序,利用该程序,可以完成 APPLESOFT 程序之间的链接。

使用时首先将 CHAIN 程序复制到我们的工作盘上,然后在第一个程序的末尾加上以下两条语句即可:

```
PRINT CHR $ (4);"BLOAD CHAIN, A520"
CALL 520"第二个程序名"
```

例如,我们将上例中的两个程序按新要求改正如下,并分别取名为 AA 和 AB,保存到工作盘上。

程序 AA:

```
5   DIM A(10)
10  FOR I = 1 TO 5: A(I) = I * 2:  PRINT A(I),
```

```
    NEXT I: PRINT
20  PRITN CHR $ (4); "BLOAD CHAIN, A520"
30  CALL 520 "AB"
```

程序 AB:

```
10  FOR J = 1 TO 5: A(J) = A(J)+ J:
    PRINT A(J),: NEXT J: PRINT
20  END
```

打入 RUN AA 命令后,屏幕将显示:

```
]RUN
2          4          6
8          10

3          6          9
12         15
```

第四章 DOS 3.3 操作系统的实用程序

4.1 DOS 3.3 系统主盘

DOS 3.3 系统主盘是一张很特别的系统引导盘,它可以在主存为 16 ~ 48KB 的 Apple Ⅱ 机上运行,也可以在主存为 64KB 的中华学习机、Apple Ⅱ e 等机器上运行。它除了含有 DOS 3.3 磁盘操作系统外,在它的软盘上还含有许多实用

程序。使用这些程序，可以能够运行整数 BASIC 程序，进行软盘片的复制，13 扇区软盘的引导，主系统盘的建立等等。此外，盘上还有不少与 DOS 及 BASIC 程序的使用有关的示范程序等。

下面给出该软盘的目录清单及简单的注释：

DISK VOLUME 254

* A	008	HELLO	系统启动后执行的第一个程序
* I	018	ANIMALS	猜字游戏程序
T	003	APPLE PROMS	由 RANDOM 程序建立的随机文件
* I	006	APPLESOFT	装入 APPLESOFT 解释程序的实用程序
* I	026	APPLEVISION	动画图形表演程序
* I	017	BIORHYTHM	计算人体生物曲线的游戏程序
* B	010	BOOT13	13 扇区软盘的引导程序
* A	006	BRIAN'S THEME	高分辨率作图示范程序
* B	003	CHAIN	APPLESOFT 链接程序
* I	009	COLOR DEMO	彩色示范程序
* A	009	COLOR DEMOSOFT	彩色示范程序
* I	009	COPY	软盘复制程序
* B	003	COPY.OBJ 0	软盘复制程序所需用到的 RWTS
* A	009	COPYA	软盘复制程序

* A	010	EXEC DEMO	EXEC 命令的示范程序
* B	020	FID	软盘文件管理程序
* B	050	FPBASIC	APPLESOFT 解释程序，由 APPLESOFT 程序装入
* B	050	INTBASIC	整数 BASIC 解释程序，由 HELLO 程序装入
* A	003	MAKE TEXT	建立文本文件的示范程序
* B	009	MASTER CREATE	建立主系统盘的实用程序
* B	027	MUFFIN	将 13 扇区的文件转换成 16 扇区文件的实用程序
* A	051	PHONE LIST	存储电话号码的应用程序
A	010	RANDOM	使用随机文件的库存管理示范程序
* A	013	RENUMBER	对 APPLESOFT 程序可重新编行号的实用程序
* A	039	RENUMBER INSTRUCTIONS	RENUMBER 实用程序的使用介绍
* A	003	RETRIEVE TEXT	取文本文件的实用程序

4.2 HELLO

HELLO 程序是用 DOS 3.3 系统主盘启动后装入运行的第一个 BASIC 程序。该程序将检查主机的内存，如果是已

扩充到 64KB RAM 主机系统，则通过装入 INTBASIC 文件，把整数 BASIC 解释程序和小汇编程序装入内存。这样系统便可以使用 INT 命令来进入整数 BASIC 状态。

在运行 HELLO 程序后，屏幕上将出现下列信息：

DOS VERSION 3.3　　　　08／25／80
APPLE II PLUS OR ROMCARD SYSTEM MASTER
(LOADING INTEGER INTO LANGUAGE CARD)

通过第二行信息，我们知道主机的内存已扩充为 64KB。而第三行信息则告诉我们整数 BASIC 解释程序已装入内存。

对于中华学习机或 APPLE Ⅱ e 机器来说，主机内存固化的是 CEC-BASIC 解释程序，要使用整数 BASIC 语言，则必须要装入整数 BASIC 程序，启动 DOS 3.3 系统主盘，或运行 DOS 3.3 系统主盘上的 HELLO 程序，是装入整数 BASIC 解释程序的最简单方法之一。

4.3　APPLESOFT

该程序是为早期固化了整数 BASIC 解释程序的 APPLE Ⅱ机用户设计的，其作用是通过将 FPBASIC 2 进制文件装入扩充为 64KB 的机器中，使得该机器可以运行 APPLESOFT 程序。对于 APPLE Ⅱ + 以后的机器，以及中华学习机等，由于都固化了 APPLESOFT 解释程序，所以该程序一般不被使用。

4.4 COPYA 和 COPY

COPYA 和 COPY 程序是软盘片复制程序，它们分别是用 APPLESOFT 语言和整数 BASIC 语言编写的，其功能和操作方法完全相同。它们都需要通过装入 COPY.OBJ0 2 进制文件来完成软盘片的复制工作。

在盘片的复制过程中，被复制的盘片称为“源盘”，而复制后的盘片称为“复制盘”。复制的最终目标是把“源盘”上的全部信息，一道一道地抄写到“复制盘”上，这样得到的复制盘是与源盘毫无区别的副本。为了防止 DOS 3.3 系统主盘在使用中被破坏，我们建议用户在使用前，一定要复制一个副本，保存起来。

软盘片的复制过程如下：

(1) 将 DOS 3.3 系统主盘插入软盘驱动器，打入命令：RUN COPYA (或 RUN COPY)

(2) 屏幕上将给出信息：

APPLE DISKETTE DUPLICATION PROGRAM

ORIGINAL   SLOT: DEFAULT=6

此时，依次回答源盘所在驱动器的槽口号和设备号分别为 6 和 1。

(3) 屏幕上将紧接着询问复制盘所在的驱动器的槽口号和设备号。

DUPLICATION   SLOT: DEFAULT=6

如果我们是在单盘驱动系统下工作的，回答仍依次为 6 和 1。

如果我们是在多盘驱动系统下工作的，则回答复制盘所

在驱动器的槽口号和设备号。

(4)全部应答完毕后，屏幕将呈现如下状态：

APPLE DISKETTE DUPLICATION PROGRAM

ORIGINAL SLOT: 6

DRIVE: 1

DUPLICATE SLOT: 6

DRIVE: 1

--PRESS 'RETURN' KEY TO BEGIN COPY--

(5) 按 RETURN 键，使开始进行盘片的复制。

如果是在单盘驱系统下，程序将告诉我们何时插入源盘，何时插入复制盘。

如果是在多盘驱系统下，程序也会通过屏幕显示告之我们程序执行到那一步了，在做什么操作。

(6) 盘片复制完毕后，主机将显示信息：

DO YOU WISH TO MAKE ANOTHER COPY?

如果需要则打 Y 键，否则按 N 键退出该程序执行。

这里值得注意的是：

(1) 盘片复制过程中，如果没有适时地插入源盘，或者盘驱动器门没关上，或者源盘片是已被加密的，或者源盘中已有文件被损坏，那么可能会出现信息：

* * * * * UNABLE TO READ * * * * *

(2) 如果没有适时地插入复制盘，或者复制盘上贴有写保护，或者错误地指定了复制盘所在驱动器占有的槽号和设备号，将出现如下信息：

* * * * * UNABLE TO WRITE * * * * *

(3) 为了防止操作上的失误而导致源盘片被损坏，建议用户在执行复盘操作前，对源盘片贴上写保护。

4.5 FID

FID 程序是用于磁盘文件管理的实用程序，它通过菜单选择对指定的磁盘文件进行有关操作。

当程序要求我们提供一个文件时，我们可以输入一个确定的文件名。也可以使用符号“=”作为“百搭”字符，字符“=”可以代替文件名中的任何字符串。例如：使用文件名“AB=CD”，程序将对所有以“AB”开头以“CD”结尾的文件名代表的文件执行操作。使用文件名“=N=”，则选择所有文件名中包含字符“N”的文件名执行操作，使用文件名“=”则选择对所有文件执行操作。

如果回答文件名时使用了百搭字符，程序会提问：DO YOU WANT PROMPTING?

(1) 如果回答 Y，则表示程序在选择到与给出的文件名相匹配的文件名时，将显示该文件名，并等待我们去证实，如果对该文件执行操作，打入 Y 键，否则打入 N 键。如果中止后继续操作则打入 Q 键。

(2) 如果回答 N，则表示程序将选择对所有与给出的文件名相匹配的文件执行操作。

该程序的执行方法：

打入命令：BRUN FID，屏幕上将显示选择菜单：

```
* * * * * * * * * * * * * * * * * * * * *
*        APPLE ][ FILE DEVELOPER        *
*                                       *
*            FID VERSION M              *
*                                       *
*  COPYRIGHT 1979 APPLE COMPUTER INC.   *
* * * * * * * * * * * * * * * * * * * * *
```

CHOOSE ONE OF THE FOLLOWING OPTIONS

(1) COPY FILES
(2) CATALOG
(3) SPACE ON DISK
(4) UNLOCK FILES
(5) LOCK FILES
(6) DELETE FILES
(7) RESET SLOT & DRIVE
(8) VERIFY FILES
(9) QUIT

WHICH WOULD YOU LIKE?9

上述菜单中 1 ～ 9 为各种不同的操作，可以通过键入相应的数字、并按回车键来选择所需的操作。

(1) COPY FILES

将源盘上的指定文件复制到复制盘上，它与 COPYA 程序所不同的是：它仅对指定文件进行复制，而不是对整张软盘进行复制。复制盘必须是经过初始化操作的盘。

(2) CATALOG

将执行 CATALOG 列文件目录的操作

(3) SPACE ON DISK

用于检查指定盘片的自由空间

(4) UNLOCK, LOCK, DELETE, VERIFY

执行这类命令，将对所指定文件执行相应的操作。

(5) RESET SLOT & DRIVE

在执行前面几项操作时，系统仅询问一次与操作有关盘

片所在驱动器的槽口和设备号，然后系统将默认这些参数，若用户想改变驱动器的槽号和设备号参数时，可以使用该参数。此时，系统将给出下列信息：

RESET SLOT & DRIVE

DONE

PRESS ANY KEY TO CONTINUE

当再次选择有关操作命令时，系统将象第一次使用该操作一样询问有关操作的参数。

(6) QUIT

该项选择将退出 FID 程序的执行。

4.6 MUFFIN

该程序的作用是将 13 扇区软盘片上的文件通过程序转换写到 16 扇区的软盘片上。

对于在 DOS 3.2 版以前的操作系统上开发的软件，均是以 13 扇区数据格式写到软盘上的，它是不能在 DOS 3.3 操作系统下运行的，否则将给出信息：UNABLE TO READ / WRITE。要想使得它能在中华学习机上使用，必须把这些文件转换成 16 扇区格式，存放到经过 DOS 3.3 初始化操作后的软盘上，MUFFIN 程序将完成此项操作。

该程序的操作步骤如下：

(1) 插入 DOS3.3 系统主盘，打入命令：BRUN MUFFIN。屏幕上将出现信息：

```
* * * * * * * * * * * * * * * * * * * *
*    APPLE ][ DOS 3.2 TO 3.3 CONVERTER    *
*                                         *
*           MUFFIN VERSION D              *
*                                         *
*  COPYRIGHT 1979 APPLE COMPUTER INC      *
* * * * * * * * * * * * * * * * * * * *

CHOOSE ONE OF THE FOLLOWING OPTIONS

        (1) CONVERT FILES
        (2) QUIT

    WHICH WOULD YOU LIKE?2
```

(2) 打入数字 1,进行文件的转换。

(3) 回答源盘和转换后的复制盘所在驱动器的槽号和设备号。

(4) 回答系统询问所要转换的文件名,可以输入确定的文件名,也可以用百搭字符(=)让系统帮助查找相匹配的文件进行转换。

(5) 取出 DOS3.3 系统主盘,插入 13 扇区的源盘,以及 16 扇区的复制盘,通过屏幕的提示,便可完成文件的转换。

转换文件的各项操作步骤与 FID 程序的 COPY 操作基本一样。所要注意的是:复制盘一定是经过 DOS 3.3 初始化操作后的软盘。

4.7 BOOT13

该程序是13扇区软盘的引导程序，如果我们拿到的是一张在DOS 3.2操作系统上开发的软盘，而想要直接运行该盘上的软盘，可用BOOT 13程序进行引导。

该程序的操作方法如下：

插入DOS 3.3系统主盘，打入命令：BRUN BOOT13，屏幕上将出现信息：

13–SECTOR BOOT UTILITY

SLOT TO BOOT FROM (DEFAULT=6)?

此时，取出DOS 3.3系统主盘，插入13扇区软盘，并打入RETURN键，即可引导13扇区软盘进行工作。

4.8 MASTER CREATE

在DOS操作系统上用INIT命令初始化新盘片时，得到的是一张从属DOS系统盘，所谓从属系统盘是指操作系统装入地址与主机内存的大小有关。如果你是在内存为48KB的机器上初始化后得到的盘片，不能在内存为16KB或32KB的APPLE Ⅱ机上引导操作系统。而对于DOS 3.3系统主盘来说，它可以在内存为16KB ~ 64KB的所有APPLE Ⅱ机上启动运行，我们说DOS 3.3系统主盘是一张主导盘(或称主盘)。

主盘和从属盘从外表上是看不出来的，甚至在机器运行过程中也看不出来，但利用MASTER CREATE程序修改问候程序，以及通过软盘上注明的标签才可知道。

MASTER CREATE 程序具有二项功能:

(1) 把一个从属盘(它的 DOS 与内存大小有关)转换成主导盘(它的 DOS 是自定位的,可以在任何大小的系统上有效地利用内存)。

(2) 可以重新指定 DOS 操作系统引导后所装入运行的程序名称。

MASTER CREATE 程序的使用步骤:

(1) 插入 DOS 3.3 系统主盘,并打入命令:BRUN MASTER CREATE.屏幕上将出现如下信息:

DOS 3.3 MASTER–CREATE UTILITY

COPYRIGHT 1980 BY APPLE COMPUTER INC

ALL RIGHTS RESERVED.

〈NOW LOADING DOS IMAGE〉

(2)如果插入的盘片不是主导盘或不是 DOS 3.3 系统盘,屏幕将出现以下二种信息之一,换上 DOS 3.3 系统主盘,并按 RETURN 键重新引导 DOS。

情况一:

IMAGE OF DOS 3.3〈MASTER〉IS NOT AVAILABLE. CHECK INSTRUCTIONS.

INSERT A SYSTEM DISKETTE AND PRESS 〔RETURN〕TO REBOOT DOS

情况二:

UNABLE TO READ IMAGE.

INSERT A SYSTEM DISKETTE AND PRESS
〔RETURN〕TO REBOOT DOS

至此，执行 MASTER CREATE 程序失败，需要重新运行。

(3) 当屏幕上出现如下提示时，请你回答该盘片 DOS 操作系统引导后，装入运行的第一个 BASIC 程序的程序名，你可以输入执行 INIT 命令操作时的文件名，也可以是其它文件名。这样用户可以根据需要指定任意的程序作为"问候"程序。

PLEASE INPUT THE "GREETING"PROGRAM'S
FILE NAME:

(4) 此时，屏幕将出现如下信息：

REMEMBER THAT "MASTER"DOES NOT
CREATE
THE "GREETING"PROGRAM, OR PLACE IT IN
THE DISK DIRECTORY

THIS IS THE FILE NAME THAT WILL BE
PLACED WITHIN THE IMAGE:

HELLO

PLACE THE DISKETTE TO BE MASTERED IN
THE DISK DRIVE

PRESS 〔RETURN〕 WHEN READY

NOTE: IF YOU WANT A DIFFERENT FILE NAME,

PRESS〔ESC〕。

从软盘驱动器中取出 DOS 3.3 系统主盘，插入要转换的从属盘，并按 RETURN 键。

(5) 转换完毕后，程序将通过如下的提示，让你选择是继续转换其它从属盘，还是退出该程序的执行。

THE DISKETTE HAS BEEN UPDATED,

YOU MAY

REMOVE IT AT THIS TIME.

IF YOU WISH TO "MASTER" ANOTHER

DISKETTE, PRESS〔RETURN〕.

OTHERWISE PRESS〔ESC〕TO EXIT "MASTER"

注意：当你执行第三步操作时，回答的文件名并不放在该盘片的目录中，它只是告诉软盘的 DOS，每次引导 DOS 后就运行该程序。如果回答的程序文件不在该软盘上，每次用该盘片启动后，将得到信息：FILE NOT FOUND。

附录A　DOS 3.3错误信息表

由于在命令或程序中使用DOS命令不当，会出现如下错误信息，这些错误信息的代码可由BASIC语句ERR=PEEK(222)得到。

错误代码	错误信息:	出错原因
1	LANGUAGE NOT AVAILABLE	没有装入整数BASIC解释程序，而使用了INIT命令。
2.3	RANGE ERROR	命令参数取值超出规定的范围。
4	WRITE PROTECTED	执行写操作时，软盘上贴有写保护。
5	END OF DATA	读操作中，读数据指针越过文本文件的末端。
6	FILE NOT FOUND	文件名拼错，没有找到该文件。
7	VOLUME MISMATCH	DOS命令使用中，卷号不匹配。
8	I/O ERROR	盘片被损坏，或软盘片未经过初始化，或未关闭驱动器的门。
9	DISK FULL	软盘上的文件个数超过额定数，或盘上数据已存满。
10	FILE LOCKED	试图对一个加有封锁的文件进行修改，删除，或重命名操作。

11	SYNTAX ERROR	DOS 命令,参数标识符或逗号分隔符用错。或没有启动 DOS 操作系统,而试图使用 DOS 命令。
12	NOBUFFERS AVAILABLE	同时打开的文件个数超出 MAXFILES 命令中所规定的文件个数。
13	FILE TYPE MISMATCH	软盘文件的类型与 DOS 命令所操作的文件类型不匹配。
14	PROGRAM TOO LARGE	没有足够的内存空间来装入新的程序。
15	NOT DIRECT COMMAND	该命令必须在程序中作为语句来使用。

第四部分
CEC汉 字 系 统

第 一 章
汉字系统的功能

中华学习机是具有汉字处理功能的计算机。它已将汉字功能部件做在主机上，使得主机本身具有汉字的输入、显示和打印等功能。中华学习机的汉字管理程序固化在只读存储器中，它与主机所固化的监控程序、CEC-BASIC 解释程序以及通过软盘所装入的 DOS 3.3 操作系统相衔接，使得中华学习机的基本软件都具有汉字处理功能，并构成一个较完整的汉字系统。

中华学习机的汉字功能部件可以根据用户的需要取舍安装。它相当于 APPLE Ⅱ机的汉卡所完成的功能。该汉字功能部件由国标一、二级全点阵汉字字库(二片 1 兆位 ROM)和汉字处理程序(一片 27256ROM)构成。它占用了系统的 3 号槽口，通过软件所提供的命令即可启动该汉字系统工作。

中华学习机汉字系统可向用户提供拼音、区位二种汉字输入方案，也可根据用户的需要扩充其它汉字输入方案。汉字系统的屏幕显示为：汉字每屏 17 × 10 个汉字，ASCII 字符

每屏 34 × 10 个字符。汉字打印可提供 15 种字型。在 BASIC 程序中，汉字的内码占用 3 个字节：由汉字引导符 7F 加上区码、位码构成，ASCII 字符则占用一个字节。BASIC 中的各项语句都可在汉字状态下使用(除 FLASH、PR #、IN # 之外)。

第 二 章
汉字系统的使用

2.1 汉字系统的启动与退出

在装有汉字功能部件的中华学习机上，可以通过软件命令或键盘上的命令键进入或退出汉字系统。

2.1.1 汉字系统的进入

(1) 直接键入"中文"键，适用于 BASIC、监控或 DOS 操作系统三种方式。

(2) 在 BASIC 程序或 DOS 操作系统状态下，可使用 PR #3 命令进入。(程序方式下，该命令必须尾随一个 PRINT 语句)

(3) 在监控方式下，可使用命令 3 CTRL-P 或 3 CTRL-K 进入。

当进入汉字系统时，屏幕显示为高分辨图形的第二页，内存的程序不受影响，但 HIMEM: 指针将指向内存地址 $ 9200。

2.1.2 汉字系统的退出

(1) 直接键入"西文"键,适用于 BASIC、监控或 DOS 操作系统三种方式。

(2) 在 BASIC 程序中,可执行命令 TEXT 或 PRINT CHR $ (17)。

(3) 在监控方式下,可以使用命令 C33AG。

退出汉字系统后,屏幕显示为文本的第一页,内存的程序不受影响,HIMEM: 指针仍指向内存地址 $ 9200。

2.2 汉字的输入方法

2.2.1 输入方法的选择

进入汉字系统后,系统处于字母输入方式,这时通过键盘输入的是 ASCII 字符。通过按下"F1"—"F5"功能键可以选择不同的汉字输入方式。功能键的定义如下:

"F1" (CTRL-A)——字母方式

"F2" (CTRL-L)——拼音输入方式

"F3" (CTRL-W)——区位码输入方式

"F4" (CTRL-T)——保留给用户扩充新的输入方案

"F5" (CTRL-F)——保留给用户扩充新的输入方案

CTRL-O 键可控制状态提示符显示或不显示。

2.2.2 拼音输入法

在进入汉字系统后,按下"F2"键(或 CTRL-L)后,屏幕提示行将显示"拼音:"两字,表示进入拼音输入方式。

拼音输入方式使用了 26 个字母键作为输入码,键入码长为 1—6 码,大小写字母均可输入。输入时按声母、韵母依次键入,系统将根据输入的音节在状态提示行上显示出该音节

的所有字，包括不同声调的字。使用者可根据提示选择所需要的字。因此，只要学过汉语拼音的人都可以使用这种拼音输入方案。

当敲入第一键后，虽然只有声母，屏幕上也会提示出以该声母打头的音节中最常用的6个汉字供用户选择，此时若有要输入的汉字，直接敲入该汉字的编号即可；若没有，需要继续敲入韵母。

当敲完韵母后，屏幕的状态提示行将显示出6个同音节汉字，若这6个同音节汉字中没有所需要输入的汉字，可敲“>”键，每敲一下“>”键，屏幕就会提示出下一幕的6个同音节汉字，直到计算机发出“嘟”叫声，表示该音节的同音汉字全部提示完。

当要寻找前一幕汉字时，可敲“<”键，此时前一幕的6个汉字将显示在状态提示行上。

当屏幕状态提示行上出现要输入的汉字时，只要敲入该汉字前面的序号，该汉字就会显示在屏幕上光标位置上，且汉字内码同时输入到键盘输入缓冲区中，如若此时连续敲入相应的序号或向前向后查找的“<” “>”键，可以连续地输入有关汉字。

在输入过程中，若拼音码被敲错，可按“△”键，使得状态提示行上最后敲入的拼音字母被删除，或者按空格键，将使得所有输入的拼音字母都被删除。当状态提示行中敲入的拼音字母均被删除后，再按“△”键，将删除已键入到内存中的最后一个汉字。

2.2.3 区位码输入法

在汉字系统状态下，按“F3”键(或CTRL–W)后，屏幕状态提示行将出现“区位:”两字，表示进入区位码输入方式。

区位码输入方式是使用国家标准 GB2381-80 区位方式输入汉字。在输入汉字时，仅使用 0～9 共 10 个数字键作为汉字的键入码，通过敲入四位数字键可以把国家标准中所规定的汉字和图形字符输入到计算机中。

例如，国标中规定 16 区 01 位置上的汉字是“啊”，键入 1601 后，“啊”字就输入到计算机中了，即：

区位码	汉字
1 6 0 1	啊
区号 位号	

国家标准中规定，00 — 10 区为图形字符，16 — 55 区为国标一级汉字，56 — 87 区为国标二级汉字。每个区有 94 个汉字或字符，其位号为 01 — 94。

在使用区位码输入汉字时，除 0～9 数字键作为汉字键入码以外，键盘上的其它键均为原来的 ASCII 字符键来定义输入。

在输入汉字过程中，如果有一输入码敲错可用“△”键来删除。如果在输入作为区位码的数字键后，又输入一个数字的字符键，则系统将接收该字符键，同时，可继续输入区位码。如果所键入的区位码不在国标码所规定的范围内，系统会发出蜂鸣警告用户，拒绝接收所输入的区位码，等待用户重新键入或删除修改。

2.2.4 用户扩充输入法

在汉字系统中，“F4”、“F5”作为用户自定义的功能键，可作为用户扩充新的输入方案时的状态转换控制码。

在用户没有将新的输入方案衔接到汉字系统之前，按“F4”“F5”键将不起任何作用。

2.2.5 特殊符号的输入

在按"F2"进入拼音方式后，键入 -(减号)；=(等号)、\(斜线)可以选择输入标点符号、算术运算符号和制表线等。这些特殊符号说明如下：

按 -(减号)可输入的符号有：

，　．　、　；　？　！　：　"　"　'　'　…

—　·　〔　〈　〕　〉　《　「　『　》　」　』

〖　【　｛　〗　】　｝

按 =(等号)可输入的符号有

￥　%　+　－　×　÷　±　＝　≈

>　<　＞　＜　／　（　〔　）　〕　∑　～

①　②　③　④　⑤　⑥　⑦　⑧　⑨　⑩

按 \(斜线)可输入的符号有：

─　│　└　┘　┌　┐　├　┤　┴　┬　┼

2.3 汉字字串与ASCII字串的处理

进入汉字系统后，键入的 ASCII 字符占用一个字节。其代码为 $ 01 ~ $ 7D。

键入的汉字均以等长的 3 字节码保存在主机的内存中。其格式为：7F+ 区码 + 位码。

由于在 BASIC 程序中，所有的字符串在存储时最高位被屏蔽，其代码值在 $ 00 ~ $ 7F 之间，同时，在 DOS 操作系统下，从软盘上读入数据时，需要使用逗号(,)和冒号(:)的代码来区分变量和语句，因此，这里的区位码不能完全和国标中的代码一样。这里的区位码是通过转换的，它与国标中规定的区位码如表 4.2.1 所示。

表 4.2.1 区位码、国标码、内码对照表

区位码	国标码	异形国标码	学习机内码	
10进制	16进制	16进制	16进制	10进制
1	21	A1	1D	29
2	22	A2	1E	30
3	23	A3	1F	31
4	24	A4	20	32
5	25	A5	21	33
6	26	A6	23	35
7	27	A7	24	36
8	28	A8	25	37
9	29	A9	26	38
10	2A	AA	27	39
11	2B	AB	28	40
12	2C	AC	29	41
13	2D	AD	2A	42
14	2E	AE	2B	43
15	2F	AF	2D	45
16	30	B0	2E	46
17	31	B1	2F	47
18	32	B2	30	48
19	33	B3	31	49
20	34	B4	32	50
21	35	B5	33	51
22	36	B6	34	52
23	37	B7	35	53
24	38	B8	36	54
25	39	B9	37	55
26	3A	BA	38	56
27	3B	BB	39	57
28	3C	BC	3B	59
29	3D	BD	3C	60
30	3E	BE	3D	61
31	3F	BF	3E	62
32	40	C0	3F	63
33	41	C1	40	64
34	42	C2	41	65
35	43	C3	42	66
36	44	C4	43	67
37	45	C5	44	68
38	46	C6	45	69
39	47	C7	46	70
40	48	C8	47	71
41	49	C9	48	72
42	4A	CA	49	73

表 4.2.1 续

区位码	国标码	异形国标码	学习机内码		区位码	国标码	异形国标码	学习机内码	
10进制	16进制	16进制	16进制	10进制	10进制	16进制	16进制	16进制	10进制
43	4B	CB	4A	74	64	60	E0	5F	95
44	4C	CC	4B	75	65	61	E1	60	96
45	4D	CD	4C	76	66	62	E2	61	97
46	4E	CE	4D	77	67	63	E3	62	98
47	4F	CF	4E	78	68	64	E4	63	99
48	50	D0	4F	79	69	65	E5	64	100
49	51	D1	50	80	70	66	E6	65	101
50	52	D2	51	81	71	67	E7	66	102
51	53	D3	52	82	72	68	E8	67	103
52	54	D4	53	83	73	69	E9	68	104
53	55	D5	54	84	74	6A	EA	69	105
54	56	D6	55	85	75	6B	EB	6A	106
55	57	D7	56	86	76	6C	EC	6B	107
56	58	D8	57	87	77	6D	ED	6C	108
57	59	D9	58	88	78	6E	EE	6D	109
58	5A	DA	59	89	79	6F	EF	6E	110
59	5B	DB	5A	90	80	70	F0	6F	111
60	5C	DC	5B	91	81	71	F1	70	112
61	5D	DD	5C	92	82	72	F2	71	113
62	5E	DE	5D	93	83	73	F3	72	114
63	5F	DF	5E	94	84	74	F4	73	115

表 4.2.1 续

区位码	国标码	异形国标码	学习机内码		区位码	国标码	异形国标码	学习机内码	
10进制	16进制	16进制	16进制	10进制	10进制	16进制	16进制	16进制	10进制
85	75	F5	74	116	90	7A	FA	79	121
86	76	F6	75	117	91	7B	FB	7A	122
87	77	F7	76	118	92	7C	FC	7B	123
88	78	F8	77	119	93	7D	FD	7C	124
89	79	F9	78	120	94	7E	FE	7D	125

例如,汉字"啊"国标码为 1601。它在机内的表示形式为:7F 2E 1D。在 BASIC 程序中用字符串函数表示则为:A $ =CHR $ (127)+ CHR $ (46)+ CHR $ (29)。

2.4 打印机的使用

在汉字系统中,汉字的打印驱动程序是针对 MX-80 III 型打印机设计的,它要求打印机的接口为 Centronics 接口,打印机接口卡必须定义为 I/O 槽口的 1 号槽。

进入汉字系统后,可以使用以下的 BASIC 命令来控制和打印汉字。

2.4.1 设置打印方式

格式:POKE 1659,n

说明:n =0,为不打印,进入汉字系统后,打印方式自动置为 0。

n=1 ~ 15,置打印,并按系统定义的15种字型进行打印。

2.4.2 设置字间距

格式:POKE 1787,n

说明:n为所打印的ASCII字符之间的字间距,一个汉字占用二个ASCII字符的位置。n取值范围为0 ~ 255,即字间距可在0 ~ 255个点之间。进入系统后,字间距自动置为1。

2.4.3 设置行距

格式:POKE 1915,n

说明:n为打印行之间的点间距,n取值范围在0 — 255之间,即行间距可在0 ~ 255个点之间。进入系统后,字间距自动置为1。

2.4.4 设置行允许字数

格式:POKE 2043,n

说明:n为一行所允许打印的汉字个数。一个汉字占有两个字符的位置。n取值范围可在0 ~ 255之间。进入系统后,行允许字数自动置为40。

由于一行所允许打印的汉字个数与打印机本身的缓存区大小有关,一般80列打印机的缓存区为400个字节左右,所以一行允许打印的汉字个数的定义与所选择的打印方式以及设定的字间距有关。

2.5 屏幕编辑命令

屏幕编辑功能和监控程序所提供的完全相同。通过键盘键入的有关命令可以移动光标所在位置,而不改变键盘输入

缓冲区中的内容。这些编辑命令以“ESC”键为第一键，以一个字母键为第二键。若第二键为I,J,K和M,则光标移动后仍处于编辑状态，即I,J,K和M可连续使用，直到按下一个非编辑功能键。若第二键为A,B,C,D,E,F,@则光标移后便自动退出编辑状态。下面列出所有编辑命令:

ESC A　光标右移一格，退出ESC状态;

ESC B　光标左移一格，退出ESC状态;

ESC C　光标下移一格，退出ESC状态;

ESC D　光标上移一格，退出ESC状态;

ESC E　从光标清至行末，退出ESC状态;

ESC F　从光标清至页末，退出ESC状态;

ESC @　清除屏幕，光标为(0,0)，退出ESC状态;

ESC I　光标上移一格，仍保持ESC状态;

ESC J　光标左移一格，仍保持ESC状态;

ESC K　光标右移一格，仍保持ESC状态;

ESC M　光标下移一格，仍保持ESC状态。

2.6　输出字符控制命令

中华学习机除了提供上述键盘方式的屏幕编辑命令外，还提供了一些程序方式的屏幕编辑命令和输出字符控制命令。它们的功能见表4.2.2。

例如，当在程序中希望清屏幕时，可用下面语句:

```
10 PRINT CHR $ (12)
```

当引导DOS操作系统后，KSW和CSW将分别指向DOS的程序地址: $ 9E81和$ 9EBD。而在DOS程序内部还设有一对I／O寄存器地址($ AA53～$ AA56)分别指

向输入／输出管理程序的入口。在文本显示方式下，KSW和CSW分别指向监控的$ FD1B和$ FDF0。而进入汉字系统后，则分别指向汉字管理程序的入口地址：$ C303和$ C32B。

表4.2.2 编辑与输出字符控制命令功能表

DEC	HEX	命令功能
CHR $ (7)	$ 07	扬声器"嘟"叫一声
CHR $ (8)	$ 08	光标退一格
CHR $ (11)	$ 0B	从光标清至页末
CHR $ (12)	$ 0C	清屏幕，光标为(0,0)
CHR $ (13)	$ 0D	输出回车符(CR)，光标移至下一行
CHR $ (14)	$ 0E	置显示方式为正常方式
CHR $ (15)	$ 0F	置显示方式为反相方式
CHR $ (17)	$ 11	退出汉字系统
CHR $ (18)	$ 12	显示或清除状态提示字符
CHR $ (19)	$ 13	暂停输出，按任一键恢复输出
CHR $ (26)	$ 1A	从光标清至行末

通过上述的讨论，我们知道：汉字管理程序可以通过KSW和CSW寄存器与固化程序相衔接，可以支持监控程序、CEC-BASIC程序和DOS操作系统具有汉字的处理功能。

第 三 章
汉字系统的子程序调用

3.1 汉字系统的实现方法

中华学习机(CEC－2000 型)利用了主机硬件所提供的辅助存储器来实现汉字系统的扩充。其汉字字库、汉字管理程序均不占用用户的程序区，它与用户程序之间通过 I／O 扩充槽口的 $ C300～$ C3FF 进行连接，实现汉字的输入、显示和打印等等。

我们知道，在监控管理程序中，I／O 操作是通过监控程序的 KSW($ 36，$ 37)和 CSW($ 38，$ 39)与键盘输入和屏幕输出的管理程序联系在一起的，通过 KSW，CSW 进入并执行的。因此，要进入汉字系统就是要将这二个跳转向量指向汉字的输入和输出管理程序。

在没有装入 DOS 操作系统时，KSW 和 CSW 分别指向 $ FD1B(键盘输入程序)和 $ FDF0(屏幕输出程序)，进入汉字系统后，KSW 和 CSW 则分别指向汉字的输入／输出程序入口 $ C303 和 $ C23B。

3.2 内部子程序调用

在汉字管理程序中，有许多子程序，用户可以通过直接调

用的方法来实现汉字的输入、汉字的输出，以及扩充汉字的输入方法等。这里仅举例说明在＄C300～＄C3FF一段的子程序入口地址及其使用方法。

3.2.1 CSWA(＄C32B)

该程序将完成ASCII字符及汉字的输出，它要求在A寄存器中存放的是字符或汉字的显示码以及显示控制命令码。对于汉字必须采用三字节的机内码，需要分三次调用才能输出一汉字。

例一：下页这段程序(程序一)中将输出信息：中华学习机COMPUTER。

3.2.2 GB.CSWA(＄C322)

该程序所完成的功能和CSWA完全一样，但它要求汉字代码采用二字节、最高位为1的国标码(即异形国标码)，需要分二次调用才能输出一汉字。ASCII字符显示码最高位必须为0。

例二：程序二将显示信息：中华学习机COMPUTER。

3.2.3 GETLN(＄FD6F)

这是监控程序中的行输入子程序，它将通过KSW间接地调用汉字的KSWA(＄C303)，是用户得到键入汉字的程序入口。

例三：在程序三中将允许用户输入一串字符，并以回车作为结束标志。程序执行完时将会把所输入的内容显示在屏幕上。

3.2.4 ZT.XS1(＄C36E)

该子程序的功能是调用汉字系统的显示状态字子程序。将系统当前的状态显示到状态行上。具体的使用方法可参见用户汉字输入方法的扩充。

程序一.

```
SOURCE  FILE:  CALL - CSW1
0000 :            1  *   CALL.CSW1
0000 :            2  * * * * * * * * * * * * * * * * * * * * *
- - - - NEXT  OBJECT  FILE  NAME  IS  CALL - CSW1.OBJ0
1000 :            3                ORG     $1000
C328 :            4  CSWA          EQU     $C328
1000 :            5  * * * * * * * * * * * * * * * * * * * * *
1000 : A9 0D      6  CALL.CSW1     LDA     #$0D
1002 : 20 2B C3   7                JSR     CSWA
1005 : A2 00      8                LDX     #$00
1007 : BD 18 10   9  LOOP1         LDA     MESAGE,X
100A : 48        10                PHA
100B : 20 2B C3  11                JSR     CSWA
100E : E8        12                INX
100F : 68        13                PLA
```

程序一续:

```
1010 : 10  F5        14                BPL    LOOP1
1012 : A9  0D        15                LDA    #$0D
1014 : 20  2B  C3    16                JSR    CSWA
1017 : 60            17                RTS
1018 : 7F  55  4F    18   MESAGE       DFB    $7F, $55, $4F, $7F, $39, $27
101B : 7F  39  27
101E : 7F  50  24    19                DFB    $7F, $50, $24, $7F, $4E, $2E
1021 : 7F  4E  2E
1024 : 7F  39  79    20                DFB    $7F, $39, $79
1027 : 20  43  4F    21                DCI                COMPUTER"
102A : 4D  50  55
102D : 54  45  D2
1030 :               22 * * * * * * * * * * * * * * * * * * * * *

* * * SUCCESSFUL  ASSEMBLY:  NO  ERRORS
```

程序二:

```
SOURCE   FILE:   CALL - CSW2
0000 :                1 *   CALL.CSW2
0000 :                2 * * * * * * * * * * * * * * * * * * * * *
---- NEXT   OBJECT   FILE   NAME   IS   CALL - CSW2.OBJ0
1000 :                3                 ORG       $1000
C322 :                4  GB.CSWA        EQU       $C322
1000 :                5 * * * * * * * * * * * * * * * * * * * * *
1000 : A9  0D         6  CALL.CSW2  LDA    #$0D
1002 : 20  22  C3     7                 JSR       GB.CSWA
1005 : A2  00         8                 LDX       #$00
1007 : BD  15  10     9  LOOP2          LDA       MESAGE,X
100A : 48            10                 PHA
100B : 20  22  C3    11                 JSR       GB.CSWA
100E : E8            12                 INX
100F : 68            13                 PLA
```

程序二续:

```
1010 : C9  0D         14                  CMP     #$0D
1012 : D0  F3         15                  BNE     LOOP2
1014 : 60             16                  RTS
1015 : D6  D0  BB     17   MESAGE         DFB     $D6, $D0, $BB, $AA, $D1, $A7
1018 : AA  D1  A7
101B : CF  B0  BB     18                  DFB     $CF, $B0, $BB, $FA
101E : FA
101F :                19                  MSB     OFF
101F : 20  43  4F     20                  ASC                 COMPUTER'
1022 : 4D  50  55
1025 : 54  45  52
1028 :                21                  MSB     ON
1028 : 0D             22                  DFB     $0D
1029                  23 * * * * * * * * * * * * * * * * * * * * * *

* * * SUCCESSFUL  ASSEMBLY:  NO  ERRORS
```

程序三:

```
SOURCE  FILE:  CALL - KSW
0000 :              1 *   CALL.CSW
0000 :              2 * * * * * * * * * * * * * * * * * * * * *
- - - - NEXT  OBJECT  FILE  NAME  IS  CALL - KSW.OBJ0
1000 :              3              ORG     $1000
FD6F                4  GETLN       EQU     $FD6F
C32B:               5  CSWA        EQU     $C32B
0200 .              6  IN          EQU     $0200
1000 :              7 * * * * * * * * * * * * * * * * * * * * *
1000 : A9 0D        8  CALL.KSW  LDA  #$0D
1002 : 20 2P C3     9              JSR     CSWA
1005 : A9 41       10              LDA     #$41
1007 : 20 2B C3    11              JSR     CSWA
100A : A9 3D       12              LDA     #$3D
```

程序三续:

```
100C:20  2B  C3      13                JSR     CSWA
100F:20  6F  FD      14                JSR     GETLN
1012:A2  00          15                LDX     #$00
1014:BD  00  02      16   LOOP3        LDA     IN,X
1017:48              17                PHA
1018:20  2B  C3      18                JSR     CSWA
101B:E8              19                INX
101C:68              20                PLA
101D:C9  8D          21                CMP     #$8D
101F:D0  F3          22                BNE     LOOP3
1021:60              23                RTS
1022:                24 * * * * * * * * * * * * * * * * * * * *

* * * SUCCESSFUL  ASSEMBLY:  NO  ERRORS
```

3.2.5 ZT.XS2($ C377)

功能为显示键入提示字符。具体用法参见用户汉字输入方法的扩充。

3.2.6 ZT.XS3($ C380)

功能为显示汉字提示。具体用法参见用户汉字输入方法的扩充。

3.2.7 USR.DCOD($ C389)

功能为将异形国标码(即国标码最高位置1)转换为学习机内码。入口、出口参数都放在A寄存器中。参见用户汉字输入方法的扩充。

3.2.8 USR.ECOD($ C392)

功能为将学习机内码转换为异形国标码。入口、出口参数都在A寄存器中。

3.2.9 BACKSP($ C39B)

功能为删除光标处字符,并使光标退一格。

3.2.10 SLECTA1($ C3A4)

功能为选择辅存1。

3.2.11 SLECTA2($ C3AB)

功能为选择辅存2。

3.2.12 SLECTM1($ C3B2)

功能为选择主存1。

3.2.13 SLECTM2($ C3B9)

功能为选择主存2。

3.3 汉字系统的内存分配及单元使用

中华学习机(CEC-I型)的汉字系统软件占用了辅存

ROM 空间和部分主存 RAM 的空间。其内存单元的分配情况如下:

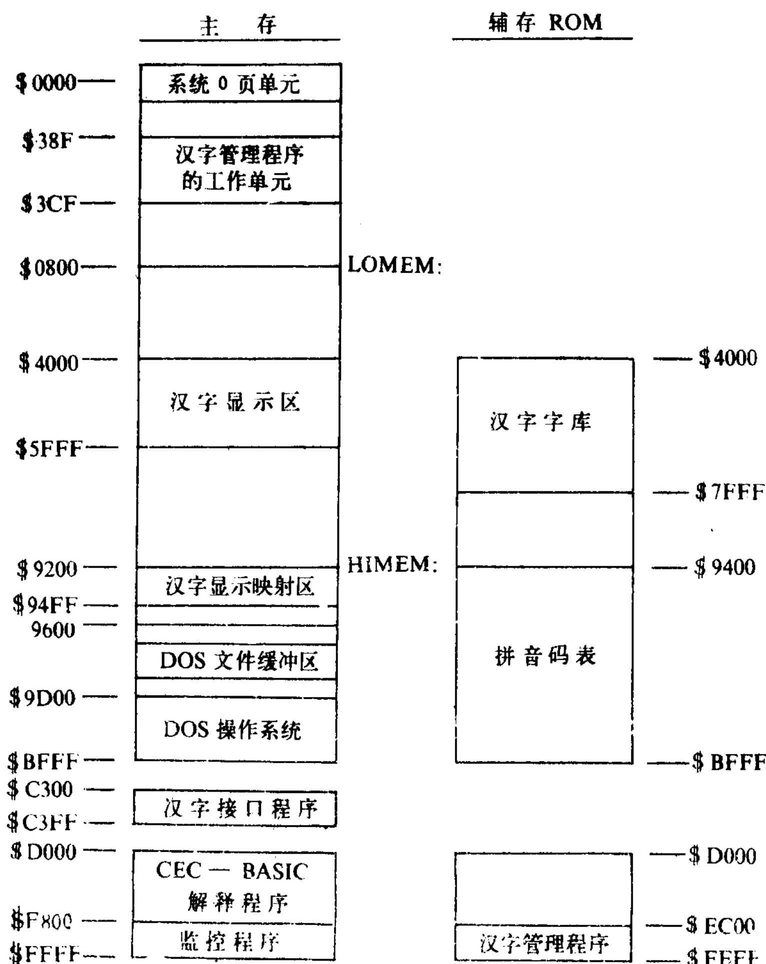


图 4.3.1

汉字系统所使用的工作单元情况如下:

0 页: $ D6、$ D7、$ ED ~ $ EF、$ F9 ~ $ FF

3 页: $ 38F ~ $ 3CF

在用户需要扩充汉字输入方式时,下述工作单元可以提供给用户使用。

单元	用途
EXPAND($ 38F)	用户扩充输入方式字符处理入口
EXPDIS($ 391)	用户扩充输入方式状态显示入口
JRP($ 399)	键入区指针
JRQSH($ 39A ~ $ 39F)	键入区
HZSH.($ 3A0 ~ $ 3A1)	汉字区首地址
HZQSH($ 3A2 ~ $ 3AD)	汉字区
FLAGC($ 3AE)	状态标志
HZP($ 3AF)	汉字区指针
SHPL.($ 3B0) SHPH.($ 3B1)	声母表指针
YMHPL.($ 3B2) YMHPH.($ 3B3)	韵母行指针
YML.($ 3B4)	韵母列指针
RJP.($ 3B5)	二级表指针
ZFQP($ 3B6)	字符区指针
RJJ.($ 3B7)	二级字计数单元
JSDY.($ 3B8)	计数单元
JSDY1.($ 3B9)	计数单元
YJCD.($ 3BA)	一级字长度
RJCD ($ 3BB)	二级字长度

DYJ.($ 3BC) 第一键标志
YJBZ.($ 3BD) 音节标志
ZFQ($ 3BE ~ $ 3BF) 字符区
JRQSH1.($ 3C0 ~ $ 3C5) 键入区 1
JRP1.($ 3C6) 键入区 1 指针
TGBZ.($ 3C7) 退格标志

其中以点结尾的变量名为拼音方式所使用。在不使用拼音方式时可提供给其他输入方式使用。不以点结尾的变量名为各种汉字输入方式共同使用,包括用户扩充的输入方式。

3.4 汉字系统的修改与扩充

3.4.1 打印机控制命令的修改

在汉字管理程序中,打印机驱动程序的设计仅适用于九针图形打印机。其型号要求必须与 EPSON MX-80 Ⅲ型打印机相兼容,如:FX-80 Ⅲ、RX-80 Ⅲ、CP-80 Ⅲ、FX-100+ 和YAMATO等型号的打印机。而对于MX-80Ⅱ、FX-80 Ⅱ、RX-80 Ⅱ等型号的打印机均不适用。

为了满足拥有上述Ⅱ型打印机,以及其它型号打印机的用户也能使用中华学习机进行汉字的打印,可以通过由用户修改打印控制命令的办法来解决

一、修改走纸命令

在 MX-80 Ⅱ型打印机上,没有 n/216 英寸走纸功能,所以必须把打印驱动程序中所使用的 ESC 3 m 命令修改成 ESC A m 命令,这样才能打印汉字。

该命令序列,我们在打印驱动程序中分别放在 LFCODE0、LFCODE2、LFCODE4(控制上半行打印走

纸)和LFCODE0、LFCODE2、LFCODE3(控制下半行打印走纸)中。在启动进入汉字系统后,我们可以选择执行下页二段程序之一,便能在 MX-80 Ⅱ型打印机上打印汉字。

注意:在执行了上述程序后,前四种打印字型不能使用,执行 POKE 1659,n 命令时,n 值必须在 5 ~ 15 之间。

二、修改打印密度

位图象打印控制命令格式为 ESC X n1n2,这里 X 可以是“K”或“L”.它们分别表示标准密度打印和倍密度打印。在汉字打印驱动程序中,我们选用的是倍密度打印(即 X 取的是字符“L”)。如果用户希望使用标准密度打印,可以通过对 TPCODE 单元的修改来完成。

在进入汉字系统后,在用户程序中只要执行语句 POKE 915,75 便可实现打印密度的转换。

无论是对走纸命令,还是对打印密度命令进行修改之后,退回到文本显示状态,再重新进入汉字系统时,只要不破坏 1147($ 47B)单元的内容,修改值是不会被破坏的。

3.4.2 用户汉字输入方法的扩充

如果用户希望扩充新的汉字输入方法,可以使用中华学习机汉字系统已提供的各种功能。

中华学习机的汉字系统提供 F4 和 F5 两个键作为进入用户扩充输入方式的功能键。

汉字系统对 F4 或 F5 键的处理如下:

当按下 F4 或 F5 键后,系统判别扩充状态显示口 EXPDIS($ 391)是否为 RTS 指令。若是,则重新读键盘,否则转入 EXPDIS 执行,显示用户提供的状态提示字,将状态标志单元 FLAGC($ 03AE)置为 F4 或 F5 的键码。F4 的键码是 CTRL-T,F5 的键码是 CTRL-F。此后,再键入的字

程序一:

```
5   REM  MX-80  II  CHANGES  LFCODE
10  POKE  916,27:  POKE  918,65:  POKE  919,8:  POKE  920,8
```

程序二:

```
SOURCE  FILE:  CHANG1
0000 :          1 *  MX-80  II  CHANGES  LF.CODE
0000 :          2 * * * * * * * * * * * * * * * * * * * * *
----NEXT  OBJECT  FILE  NAME  IS  CHANG1.OBJ0
1000 :          3              ORG      $1000
0394 :          4  LFCODE0     EQU      $394
0396 :          5  LFCODE2     EQU      $396
```

程序二续:

```
0397 :                 6  LFCODE3   EQU   $397
0398 :                 7  LFCODE4   EQU   $398
1000 : A9 1B           8            LDA   #$1B
1002 : 8D 94 03        9            STA   LFCODE0
1005 : A9 41          10            LDA   #$41
1007 : 8D 96 03       11            STA   LFCODE2
100A : A9 08          12            LDA   #$08
100C : 8D 97 03       13            STA   LFCODE3
100F : 8D 98 03       14            STA   LFCODE4
1012 : 60             15            RTS
1013 :                16 * * * * * * * * * * * * * * * * * * * * *

* * * SUCCESSFUL ASSEMBLY: NO ERRORS
```

符只要是大于等于＄ A0的码(即可显示的字符)或等于＄ 88(退格键)都转到用户扩充的输入处理程序入口地址EXPAND(＄ 38F)去执行。其他控制字符由系统程序内部处理。键入字符是通过A累加器转到用户输入处理程序的。当返回时,A累加器的内容为零,则系统继续读键;非零,则将A中内容送入键盘输入缓冲区。

根据上述系统程序的处理,用户的输入程序需要完成下述任务。

1. 进入汉字系统后,运行用户输入程序。该程序要修改EXPDIS中的内容,使其指向用户显示状态字的处理程序入口。修改EXPAND中的内容,使其指向用户输入处理程序的入口。

例如,设用户显示状态字程序入口为USER1,用户输入处理程序入口为USER2。修改指针的程序可如下编制:

```
START   LDA   # >USER1
        STA   EXPDIS
        LDA   # >USER1
        STA   EXPDIS+ 1
        LDA   # >USER2
        STA   EXPAND
        LDA   # <USER2
        STA   EXPAND+ 1
        RTS
```

2. 用户显示状态字程序只要将状态字字符区的首地址填入状态指针单元(即ZTZZ)即可。例如,上例中显示状态字程序入口为USER1,设状态字符区首地址为STATUS,状

态字为"国标 ",则显示状态字程序可如下编制:

```
USER1    LDA   # >STATUS
         STA   ZTZZ
         LDA   # <STATUS
         STA   ZTZZ+1
         JSR   ZT.XS1
         JMP   SLECTA2
STATUS   DFB   $ 7F, $ B9, $ FA (国)
         DFB   $ 7F, $ B1, $ EA (标)
         DFB   $ 3A, $ 20(:)
```

上例中状态指针单元 ZTZZ 为 $ FB、$ FC。ZT XS1 为汉字系统提供的显示状态字子程序入口。JMP SLECTA2 是将有效内存切换到辅存 2,返回。这是因为调用 ZT . XS1 后,汉字系统将有效内存切换到主存 ROM BASIC 存储区,而用户扩充的输入程序执行完某一功能是要返回汉字系统程序的,而汉字系统程序在辅存 2 中,因此,在返回系统程序前要切换到辅存 2,然后返回。

3. 用户输入程序对输入字符的处理,应首先判别 A 累加器中是否为 $ 88,即退格键。若是,则转到退格键处理。否则为输入字符处理。

4. 当用户需要将输入字符暂存时,可使用键入区 JRQSH($ 039A ~ $ 039F) 和键入区指针 JRP ($ 03B5)。对 JRQSH 中内容的存取是通过首地址(即 JRQSH = $ 039A)加位移量(JRP 的值)确定的。

5. 当用户需要在屏幕状态行上显示键入字符作为提示时,可调用 ZT . XS2($ C377)。此调用将按 JRP 的值将

JRQSH 中的字符显示到状态行上。例如，若 JRP=2，则显示 JRQSH 区中前两个字符，若 JRP=0 则清除显示提示。

6．当用户需要存放与键入的音或形有关的汉字时，可使用汉字区 HZQSH（$ 03A2～$ 03AD）和汉字区指针 H2P（$ 03AF）。对 HZQSH 中内容的访问是通过首地址（即 HZQSH）加位移量（HZP 的值）确定的。

7．当用户需要在屏幕状态行上显示汉字提示时，可调用 ZT．XS3（$ C380）。此调用将按汉字区指针 HZP 的值将 HZQSH 区中的汉字显示到状态行上。注意，HZQSH 中的汉字码为异形国标码，最多放六个。ZT．XS3 将自动在每个汉字前插入序号并显示。

8．用户需要将汉字码送入系统键盘缓冲区时，必须使用字符区 ZFQ，将学习机内码的区位码放入 ZFQ（$ 03BE）和 ZFQ+1（$ 03BF）。将字符区指针 ZFQP（$ 03B6）放入 2。然后将 $ FF 送入 A 中，切内存到辅存 2，返回。

9．若用户需要将异形国标码转换为学习机内码，可将异形国标码放在 A 中，调用 USR．DCOD（$ C389），返回时 A 中为学习机内码。若需要将学习机内码转换为异形国标码，可将学习机内码送 A，调用 USR．ECOD（$ C392），返回时 A 中为异形国标码。

10．对于退格键，系统规定为先退状态行键入的提示字符，当提示字符退完后，即 JRP=0 后，再退光标提示处的字符。用户只需将 JRP 的值减 1，再调一次 ZT．XS2，即可删除一个键入提示字符。用户也可以根据自己的需要，调用 BACKSP（$ C3），使光标处删除一个字符。

附录 A　汉字管理系统程序框图

一、汉字系统的启动程序

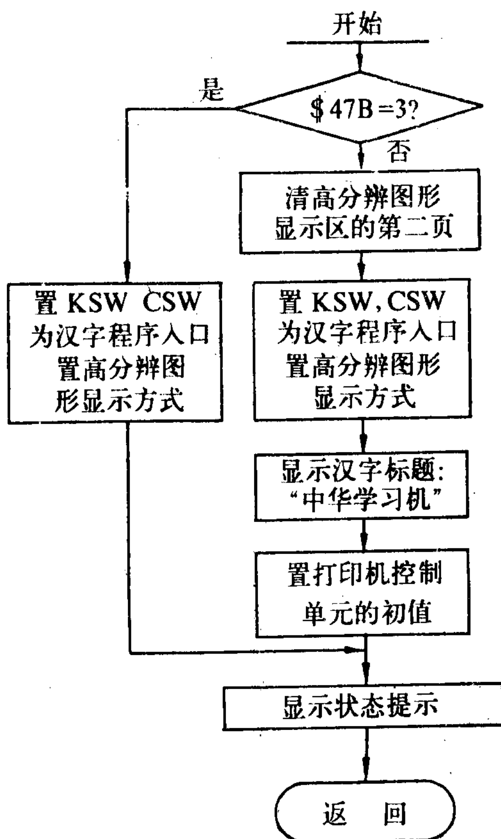


图 4.A.1　汉字系统的启动处理程序框图

二、CSW 处理程序

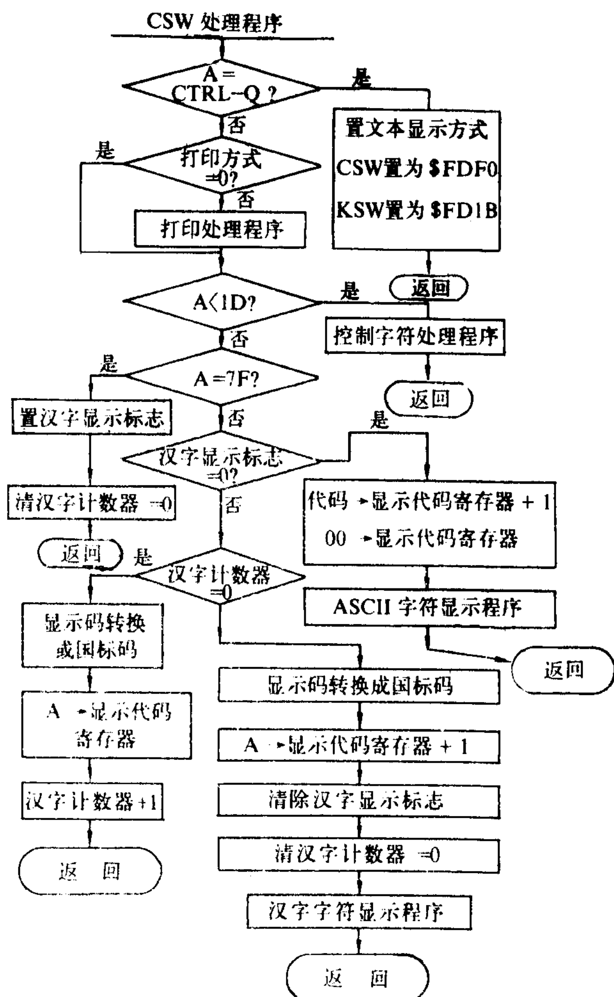


图 4.A.2 CSW 处理程序框图

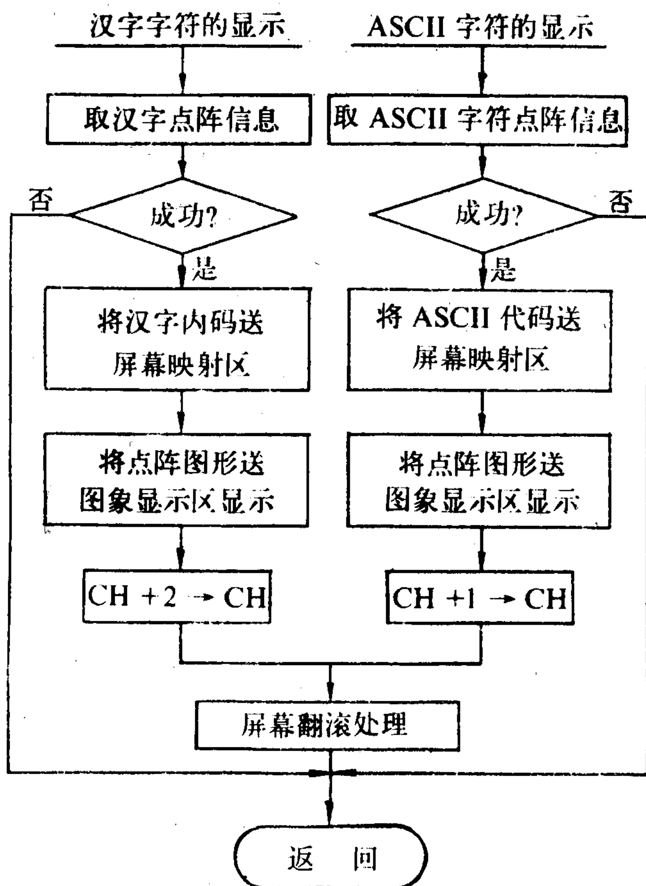


图 4.A.3　字符显示处理程序框图

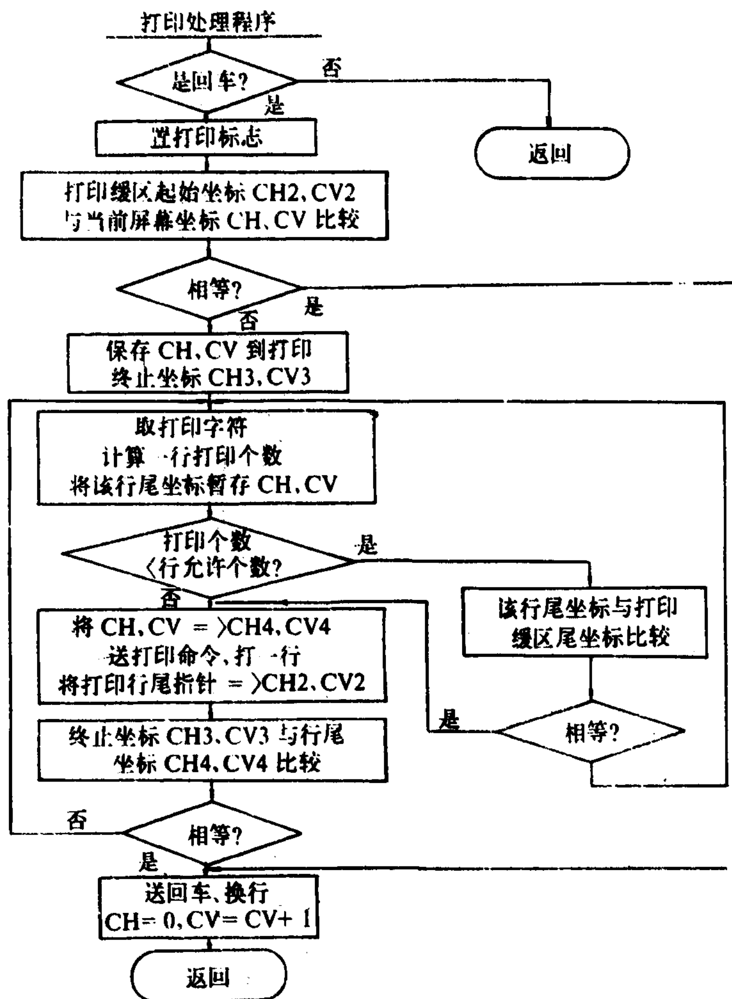


图 4.A.4　打印处理程序框图

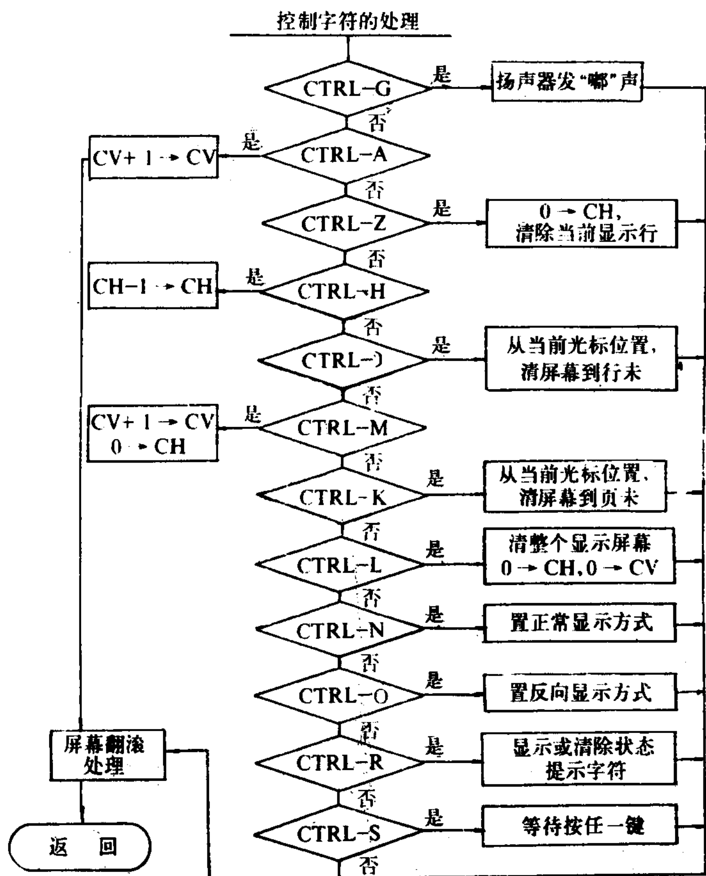


图 4.A.5　显示控制字符处理程序框图

三、KSW 处理程序

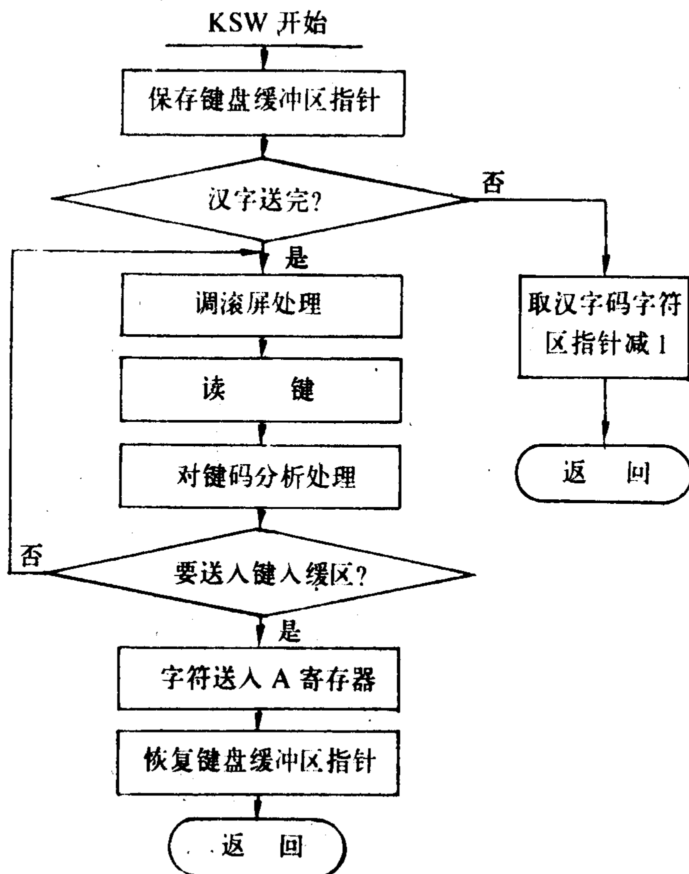


图 4.A.6　KSW 处理程序框图

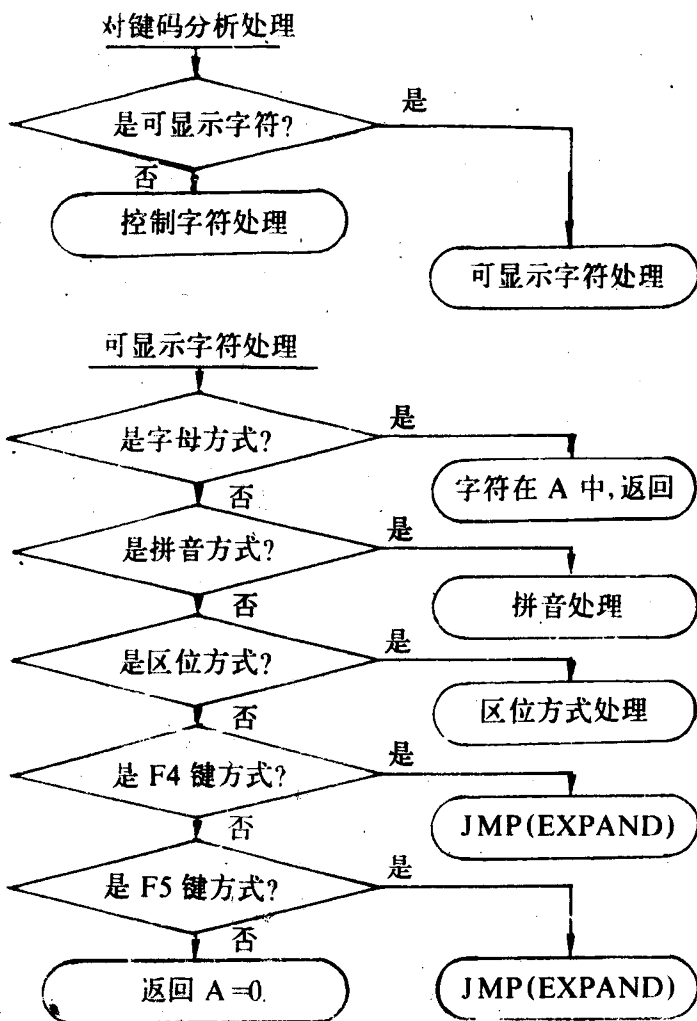


图 4.A.7 KSW 处理程序框图

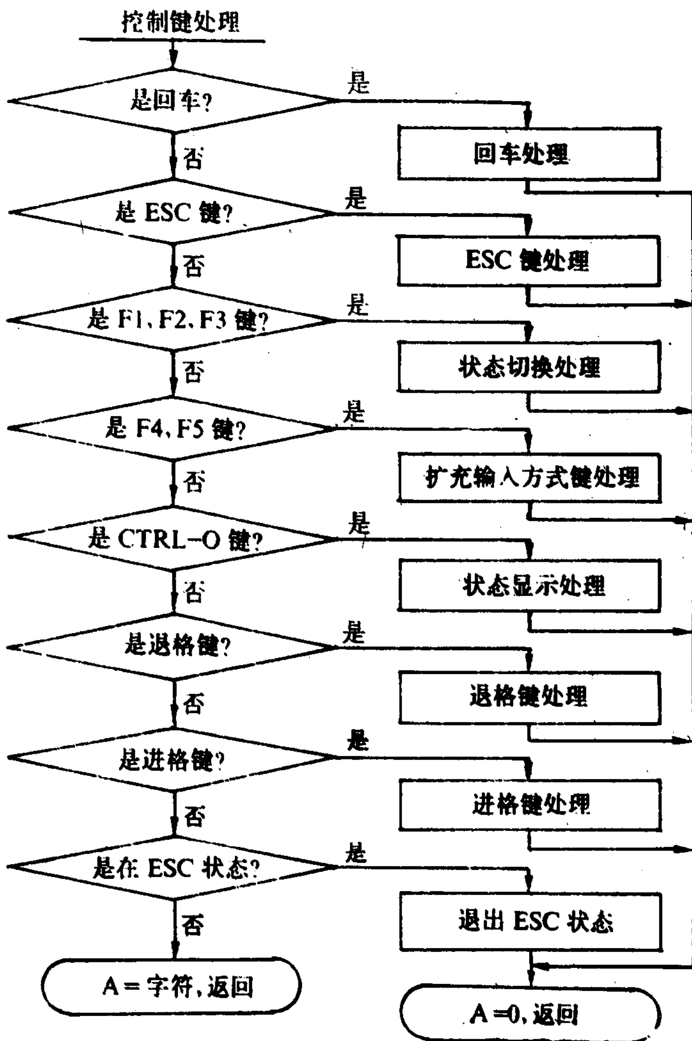


图 4.A.8　控制键处理程序框图

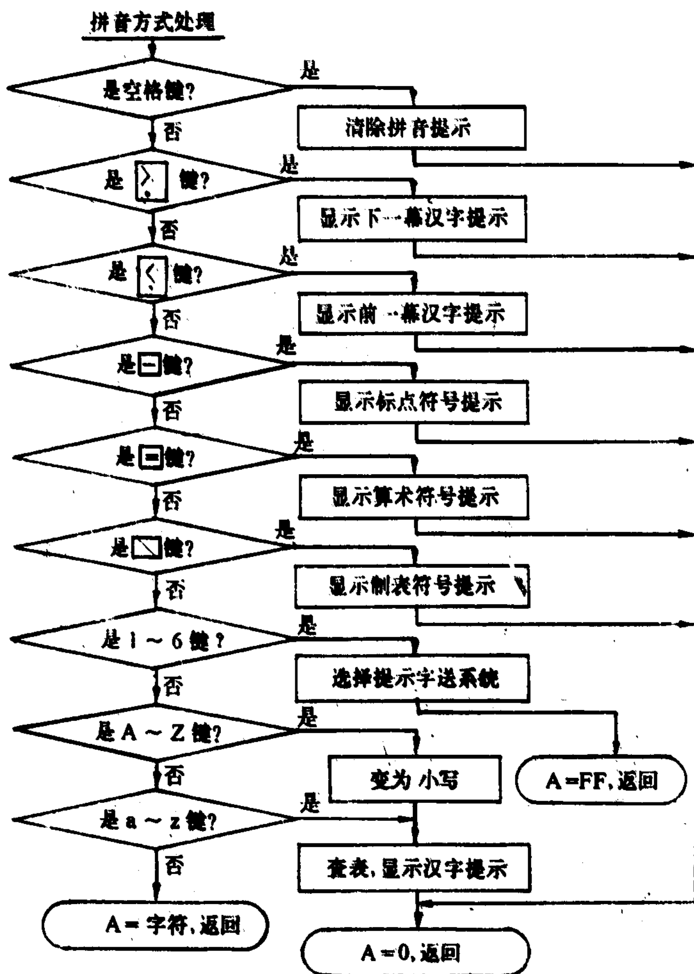


图4.A.9 拼音方式处理程序框图

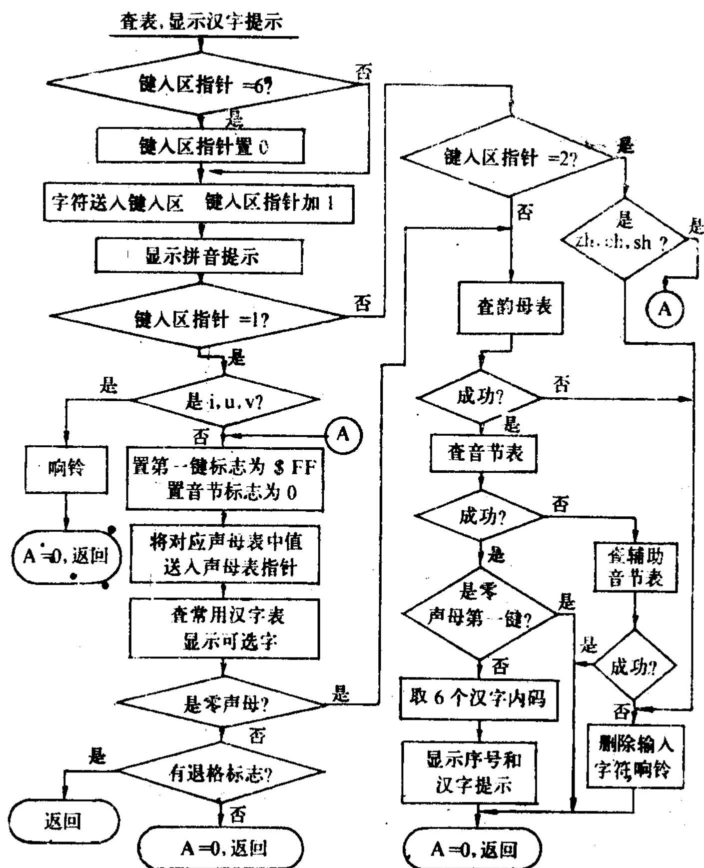


图 4.A.10　拼音方式处理程序框图

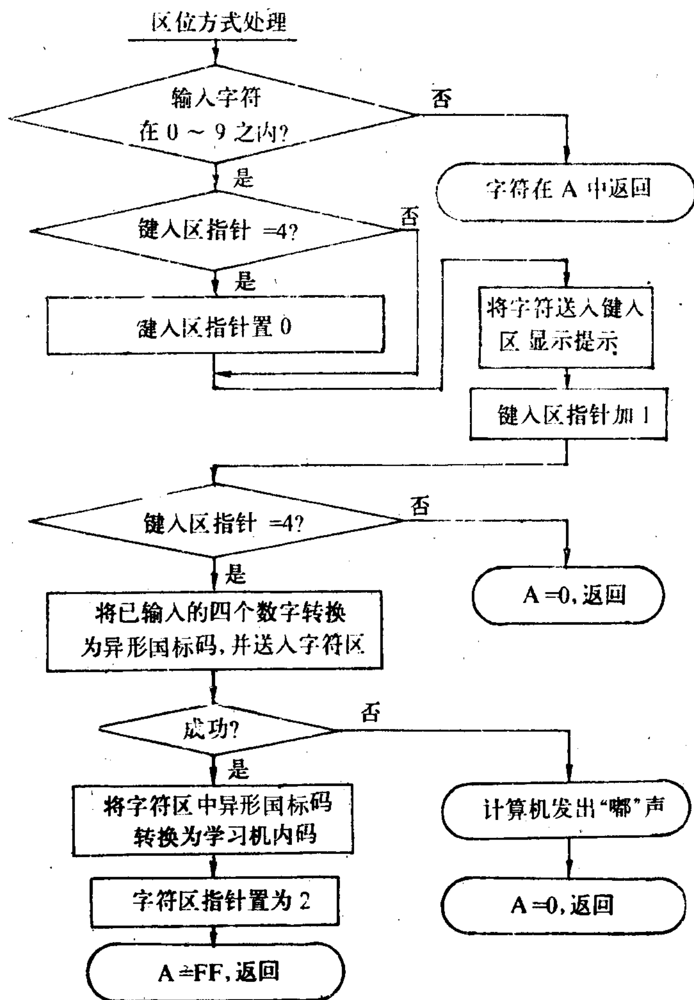


图 4.A.11 区位方式处理程序框图

第 五 部 分
DOS3.3 操作系统分析报告

第 一 章
DOS 3.3 磁盘操作系统概述

1.1 DOS3.3磁盘操作系统的结构

DOS3.3 磁盘操作系统从结构上可以分成三个主要部分。

1. 主体程序：它是 DOS 操作系统的主程序。它包括有：DOS 操作的启动入口程序，与 BASIC 的接口程序，DOS 命令解释程序。

2. 文件管理程序：磁盘文件的读写和维护管理程序。

3. 磁盘驱动程序(RWTS)：磁盘扇区的读写及格式化操作程序。

DOS 操作系统的这三个部分之间的关系如图 5.1.1 所示。它们之间具有明确的层次关系，外层程序可以通过一定的命令参数来调用内层的程序模块，反之不行。

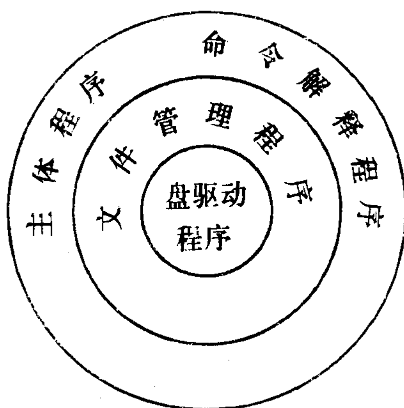


图 5.1.1 DOS 的层次结构

1.2 DOS操作系统的内存分配

DOS 3.3 磁盘操作系统是用 6502 指令编写的，当系统引导后，将被装入主机内存，整个 DOS 操作系统为常驻内存。

在经过 DOS 操作系统初始化的磁盘上，第 0 ～ 2 道上装有 DOS 操作系统程序，因此它们都可用来引导系统。DOS 操作系统装入的内存地址与用来引导的盘有关。若用系统主盘进行引导，DOS 操作系统将装入到当前系统的最高内存处，如果用从属盘进行引导，则装入到与该从属盘进行初始化时的系统相同的内存位置。因此在较大内存的系统上初始化后的盘片不能在较小内存的系统上执行引导。

以内存为 48KB 的系统为例，DOS 操作系统所占用的内存地址如图 5.1.2 所示。

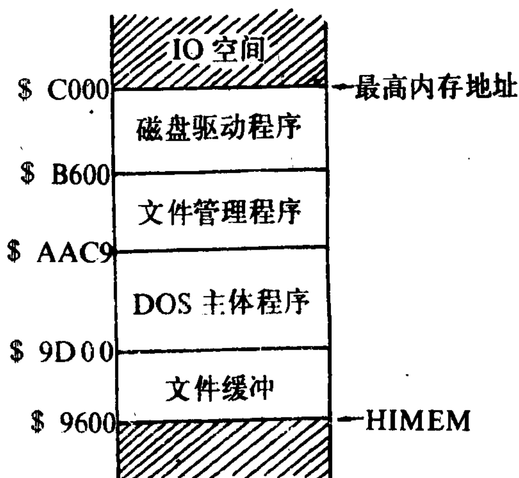


图 5.1.2　DOS 内存分配

DOS 操作系统程序占内存约 10K，其中，主要程序占 3.5K，文件管理程序占 2.8K，RWTS 程序占 2.5K。另外文件缓冲区占 1.75K。这样的文件缓冲区最多可放三个文件，每个文件的文件缓冲区为 595 个字节。

1.3　DOS 与 I／O 驱动程序的接口

输入／输出设备与主机相连接，一般均需要配有 I／O 驱动程序对它进行管理。对于主机板上提供的输入／输出设备来说，其驱动程序已包含在主机上的监控管理程序中。例如：显示器，键盘，磁带机等接口。而对于在主机扩充槽口上所连接的设备，其驱动程序在各个槽口所占用的地址空间

中，例如：软盘驱动器接口，打印机，汉字卡，通讯卡，80列显示卡等。在整个机器系统中，要有一个控制程序来完成I／O的切换和控制工作。即在系统需要进行输入／输出操作时，转入指定的I／O设备的驱动程序去执行。

1.3.1 监控程序的I／O控制

监控程序中设立了两个寄存器，一个是输入寄存器KSW（地址为$ 38, $ 39单元），另一个是输出寄存器CSW（地址为$ 36, $ 37），它们的内容指出了当前输入和输出驱动程序的入口地址。

在没有DOS操作系统的情况下，KSW指出的是监控程序中的键盘输入程序的入口地址（$ FDIB），CSW指出的是监控程序中的屏幕输出程序的入口地址（$ FDF0）。在用监控命令或BASIC命令进行I／O设备切换时（例如：PR #n, IN #n, nCTRL-K, n CTRL--P等），修改这二个寄存器，就可以指向不同的I／O驱动程序。

1.3.2 DOS操作系统下的I／O控制

在DOS操作系统装入后，监控程序的输入／输出寄存器指出的是DOS操作系统的程序地址：KSW指向DOS的输入程序的入口地址（$ 9E81），CSW指向DOS的输出程序的入口地址（$ 9EBD）。

DOS操作系统程序本身也有一对输入和输出寄存器，地址分别为$ AA53～$ AA54和$ AA55～$ AA56。由它们分别指向当前I／O驱动程序的入口地址。一般情况下，它仍然是指向监控程序中的键盘输入程序入口（$ FD18）和屏幕输出程序入口（$ FDF0）。

无论是否引导DOS操作系统，BASIC解释程序和监控程序都将根据KSW和CSW寄存器的指针转向输入／输出

操作的入口。

第 二 章
磁盘信息的组织与管理

2.1 磁盘的分配

APPLE Ⅱ / 中华学习机配置使用的是单面单密度 5.25 英寸软盘片。每张盘片在 DOS 3.3 操作系统支持下划分为 35 个磁道,每道划分为 16 扇区,每个扇区可记录 256 个字节的用户信息,所以整张软盘可储存 143KB 信息。

对于每张经 DOS 初始化操作后的软盘,第 0,1,2 道上存放的是 DOS 操作系统的副本程序。第 17 道存放的是磁盘管理信息。而其它磁道才可用来存放用户程序,所以用户实际可用的盘区为 496 个扇区,共 124KB。

系统对用户文件在磁盘上的空间分配顺序为:从第 18 道向第 35 道依次分配,然后再从第 16 道向第 3 道依次分配。这样安排的目的是要使经常访问的盘管理信息及文件信息尽量位于磁盘的中央道附近,以减少访盘时磁头移动的相对距离,以提高访盘的速度。

2.2 磁盘的管理信息

在 DOS 软盘的第 17 道上存放有部分磁盘管理信息,它

包括有VTOC和文件目录区二部分。

2.2.1 VTOC表

VTOC (Volume Table of Contents)占用一个扇区，位于17道0扇区，它给出的是磁盘的一些标志及盘区的分配情况，如表5.2.1所示。

表5.2.1 磁盘标志及盘区分配区

字节($)	内　　容
$ 00	未用
$ 01 ~ $ 02	第一个目录区段的道号和区号
$ 03	对该盘进行格式化的DOS的版本号
$ 04 ~ $ 05	未用
$ 06	盘卷号(1 ~ 254)
$ 07 ~ $ 26	未用
$ 27	在一个道／区表扇区中所能存放的道／区对的最大数(122)
$ 28 ~ $ 2F	未用
$ 30	已分配的最后一道
$ 31	盘的分配方向(+ 1 或 −1)
$ 32 ~ $ 33	未用
$ 34	每个盘的道数(35)
$ 35	每道的扇区数(13 或 16)
$ 36 ~ $ 37	每个扇区的字节数
$ 38 ~ $ 3B	第0道的自由扇区图
$ 3C ~ $ 3F	第1道的自由扇区图
·········	
$ C0 ~ $ C3	第34道的自由扇区图
$ C4 ~ $ FF	若盘上的磁道数大于35，则仍放其自由扇区图

在 VTOC 表中,对磁盘上的每一磁道都分配 4 个字节作为自由扇区的标志字节,它标出该磁道中哪些扇区已被使用。扇区分配原则是从 $ 0F 区向 $ 00 区依次进行。自由扇区图的安排如下:

字节	对应扇区
0 ~ 1	F E D C B A 9 8 7 6 5 4 3 2 1 0
2 ~ 3	未用

前二个字节中,每一位对应于一个扇区,已使用的扇区相应位为 0,未使用的扇区对应位为 1。

2.2.2 文件目录区

文件目录区放在 $ 11 道的 $ 01 ~ $ 0F 扇区中,一般从 $ 0F 扇区依次向 $ 01 扇区存放。每一个目录扇区可以存放 7 个文件项,所以一张盘最多可以存放 105 个不同的文件。

由于文件目录的第一个扇区位置由 VTOC 中的指针指出,而下一个目录扇区又是由前一个目录扇区中的链接指针所指出,所以很容易修改标准 DOS,使其文件目录区存放在软盘的其它位置,而且目录扇区的个数也可以适当增加,使得在一个盘上可以存放更多的文件。

文件目录扇区的安排如表 5.2.2 所示。

每一个文件均有一个文件描述项,它占有 35 个字节位置,文件描述项内容如表 5.2.3 所述。

文件类型及标志如表 5.2.4 所述。

当 DOS 操作系统访问某个盘文件时,首先读入 VTOC 表,从中找出第一个目录扇区的位置,从该目录扇区开始按链

接指针查找各个目录扇区，直到从中找到所需文件的文件描述项为止。从文件描述项中可得到该文件的基本情况(文件类型，文件长度，在盘上的位置等)，然后再根据磁道／扇区表(T／S 表)可找到该文件的全部信息。

表 5.2.2 文件目录扇区的安排

字节	内容
$ 00	未用
$ 01 ~ $ 02	下一个文件目录扇区的道号($ 11)，扇区号
$ 03 ~ $ 0A	未用
$ 0B ~ $ 2D	第一个文件描述项
$ 2E ~ $ 50	第二个文件描述项
⋮	⋮
$ DD ~ $ FF	第七个文件描述项

表 5.2.3 文件描述项内容

字节	内容
$ 00 ~ $ 01	第一个道／区表扇区的道号、区号
$ 02	文件类型及标志
$ 03 ~ $ 20	文件名(占 30 个字符)
$ 21 ~ $ 22	以扇区为单位表示的文件长度

表 5.2.4 文件类型及标志

未封锁	已封锁	文件类型
$ 00	$ 80	文本文件(T)
$ 01	$ 81	整数 BASIC 文件(I)
$ 02	$ 82	APPESOFT 文件(A)
$ 04	$ 84	2 进制文件(B)
$ 08	$ 88	S 型文件
$ 10	$ 90	R 型文件(EDASM 生成的)
$ 20	$ A0	新 A 型文件
$ 40	$ C0	L 型文件(新 B 型文件,由 LISA 程序建立)

2.3 磁盘文件的存储格式

2.3.1 磁道 / 扇区表

每个文件均有一相应的磁道 / 扇区表,用来指明该文件在盘上占用了哪些磁道和扇区。文件描述项中指明了某文件的磁道 / 扇区表的第一个扇区的位置。磁道 / 扇区表为 256 个字节,其格式如表 5.2.5 所示。

当访问某一文件时,从文件描述项中得到该文件的磁道 / 扇区表的位置,然后从该表中得到各文件扇区所占用的磁盘位置。当文件的长度超过 122 个扇区时,则需要分配另一个磁道 / 扇区表扇区。由于任何文件至少要有一个磁道 / 扇区表扇区,所以即便是一个空文件,它也至少占用 2 个扇区的磁盘空间。

表 5.2.5 磁道 / 扇区表

字 节	内 容
$ 00	未用
$ 01 ~ $ 02	下一个磁道 / 扇区表所占磁道号，扇区号
$ 03 ~ $ 04	未用
$ 05 ~ $ 06	该磁道 / 扇区表是本文件磁道 / 扇区表的第几个扇区
$ 07 ~ $ 0B	未用
$ 0C ~ $ 0D	第 1 个文件扇区所占用的磁道号，扇区号
$ 0E ~ $ 0F	第 2 个文件扇区所占用的磁道号，扇区号
$ FE ~ $ FF	第 122 个文件扇区所占用的磁道号，扇区号

2.3.2 文件的存储格式

从文件描述项说明中可得知：文件的类型及标志占用一个字节，除封锁标志占用最高位以外，由 7 位字节加上全 0 共可表示 8 种类型的文件，但 DOS 3.3 操作系统本身能够直接建立和处理的文件类型只有四种：文本文件（T），整数 BASIC 程序（I），APPLESOFT 程序（A）和 2 进制文件（B）。其它几种类型的文件则需要由其它程序来建立和使用。例如：L 类型文件（新 B 型文件）需要由 LISA 汇编程序来建立和使用。

1. 文本文件

文本文件一般是存放数据信息的文件。它分成随机文件和顺序文件两种。文本文件均由记录构成，每个记录之间用

CR 字符分隔,并以 $ 00 作为文件结束的标志,或以道 / 区表的指针的结束来标志文件的结束。顺序正文文件的各记录的长度是不定的,仅以 CR 作为分隔,所以信息一般以顺序方式读写。随机文件的各记录是定长的,因而读写可从任一指定的记录处开始。

文本文件中的所有信息均以可打印的 ASCII 字符形式存放,包括数字。所以可以用 $ 00 作为标志字节。也由于这一点,使得文本文件比起 2 进制文件来讲,处理速度要慢,占用的盘空间也增多(每一位数字要占一个字节)。

文本文件的存储格式为:

记录 1 CR 记录 2 CR 记录 3 …… CR $ 00

2. 2 进制文件

2 进制文件存放的是机器内存中的 2 进制信息的副本,即把内存中某一起始地址开始的一段信息保存到软盘上而生成的文件。

2 进制文件存储格式为:

地址　　长度　　内存 2 进制信息映象

其中,地址指明了该段信息原先在主机内存中存放的起始地址,长度指内存映象的字节个数。它们均占用二个字节,先放低字节,后放高字节.

3. BASIC 程序文件

在 DOS 操作系统中,把 BASIC 程序分成两种类型的文件(A 类型和 I 类型),其存储格式完全相同。

BASIC 程序文件的存储格式为:

长度　　程序的内存映象

其中,长度指的是该程序的内存映象的字节数,它占用二个字,先放低字节、后放高字节。

第 三 章

DOS 主体程序的分析

DOS 操作系统主体程序是 DOS 的主控部分，它将完成整个操作系统的装入，启动操作，与 BASIC 程序的接口，以及 DOS 命令的解释和执行。

3.1 DOS的引导与装入

DOS 的引导是将 DOS 操作系统软件从盘上装入到主机内存。DOS 操作系统软件放在软盘的 \$ 00 ~ \$ 02 道，第 \$ 00 道和 \$ 01 道占用了 16 个扇区，第 \$ 02 道占用了 5 个扇区，共 37 个扇区。

DOS 的引导和装入过程分三步完成。

3.1.1 BOOT0

当机器加电或执行 DOS 引导命令时，即执行磁盘的引导程序 BOOT0。其程序框图如图 5.3.1。

BOOT0 程序放在软盘驱动器控制接口卡上，由 PR #n 命令来启动执行。该程序的主要工作是从第 \$ 00 道 \$ 00 扇区读入 256 个字节，存放在 \$ 800 ~ \$ 8FF 中去，这段程序称为 BOOT1，然后转入 BOOT1 程序去执行。所以 BOOT0 的工作可以说是装入并运行 BOOT1 程序。

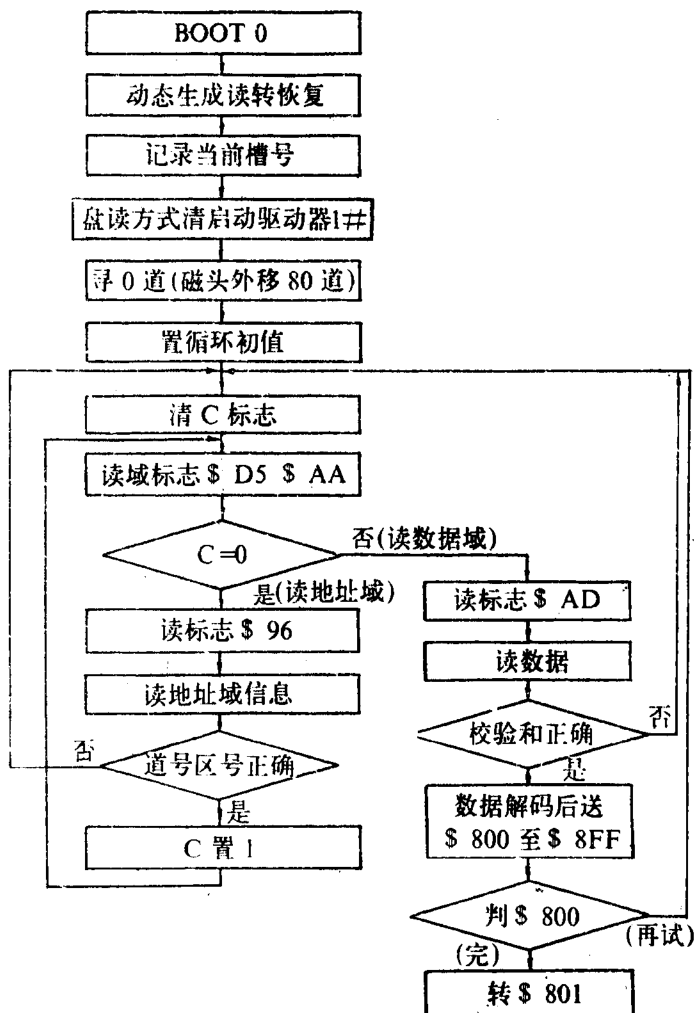


图 5.3.1 BOOT0 框图

3.1.2 BOOT1

BOOT1 程序的主要工作是装入 BOOT2 程序和 RWTS 程序，然后，转入 BOOT2 程序执行。BOOT1 框图如图 5.3.2。

BOOT1 中读扇区的工作是通过调用 BOOT0 来完成的，它将从盘上第 \$ 00 道 \$ 00 ～ \$ 09 扇区的信息装入到 DOS 的程序区中。装入的地址随主盘和从属盘而不同，对于主盘存放地址为 \$ 3600 ～ \$ 3FFF，对 48K 系统的从属盘则存放地址为 \$ B600 ～ \$ BFFF。

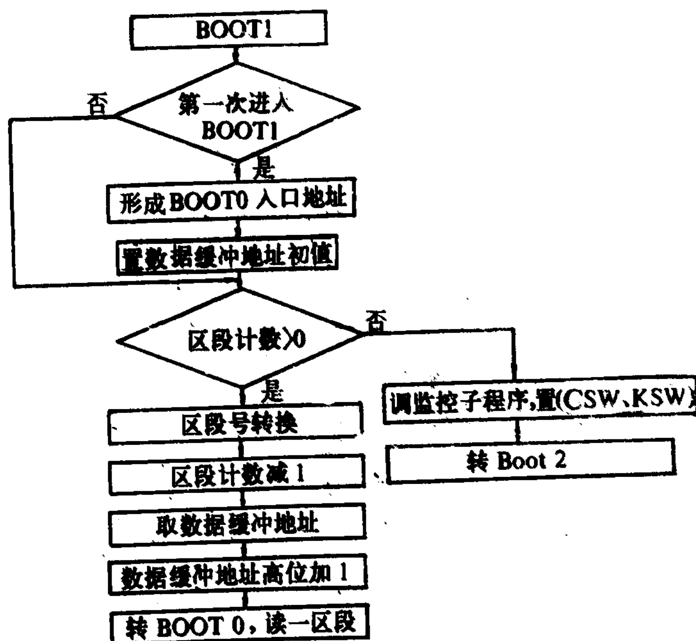


图 5.3.2　BOOT1 框图

3.1.3 BOOT2

BOOT2 程序由 BOOT1 程序装入并执行。其主要工作是:将整个 DOS 操作系统装入内存,然后转 DOS 的冷启动程序入口,参见图 5.3.3。

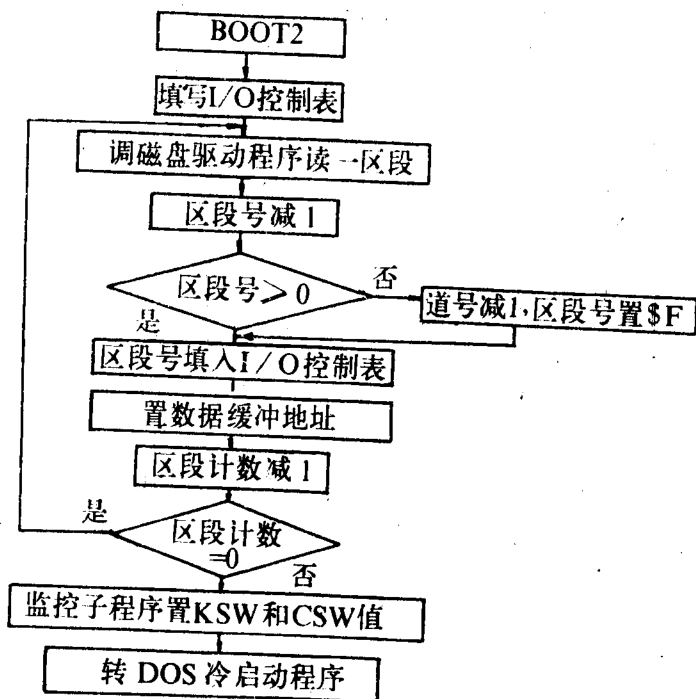


图 5.3.3 BOOT2 框图

BOOT2 程序是利用已装入的 RWTS 程序模块来读盘的。BOOT2 读入 DOS 的顺序是:先由 $ 02 道的 $ 04 扇区到 $ 00 扇区,然后是 $ 01 道 $ 0F 扇区到 $ 00 扇区,最后是 $ 00 道的 $ 0F 扇区至 $ 0A 扇区,共 27 个扇区。

对于从属盘来说,在 BOOT2 程序装入整个 DOS 之后,引导工作就算完成了。但对于主盘来说,还将执行 $ 1B00 ~ $ 1CFF 中的一段移动程序将 DOS 操作系统程序从 $ 1D00 ~ $ 3FFF 移到当前系统的最高内存位置。这段移动程序在软盘的第 $ 00 道的 $ 0A ~ $ 0B 扇区,也是由 BOOT2 程序装入的,移动操作完成后,再转入 DOS 的冷启动入口。

3.2 DOS的冷启动过程

DOS 操作系统由三级引导装入内存后,便进入冷启动过程,其入口地址为 $ 9D84。它的主要工作:初始化 DOS 中的 BASIC 语言标志,置 DOS 的启动标志,然后转入 BASIC 解释程序,即退出 DOS。图 5.3.4 为冷启动程序框图。

3.3 DOS的输入和输出程序

DOS 的输入和输出程序是 DOS 中的两个关键接口程序。它将完成 DOS 与 BASIC 的接口工作,DOS 通过对 BASIC 解释程序的输入和输出信息的截取,完成对 DOS 命令的识别、接收和处理。

在 DOS 操作系统完成冷启动操作后,监控的 I/O 寄存器已为 DOS 的输入程序和输出程序的入口所取代,一旦需要

进行输入／输出时，BASIC解释程序即把控制交给了DOS。

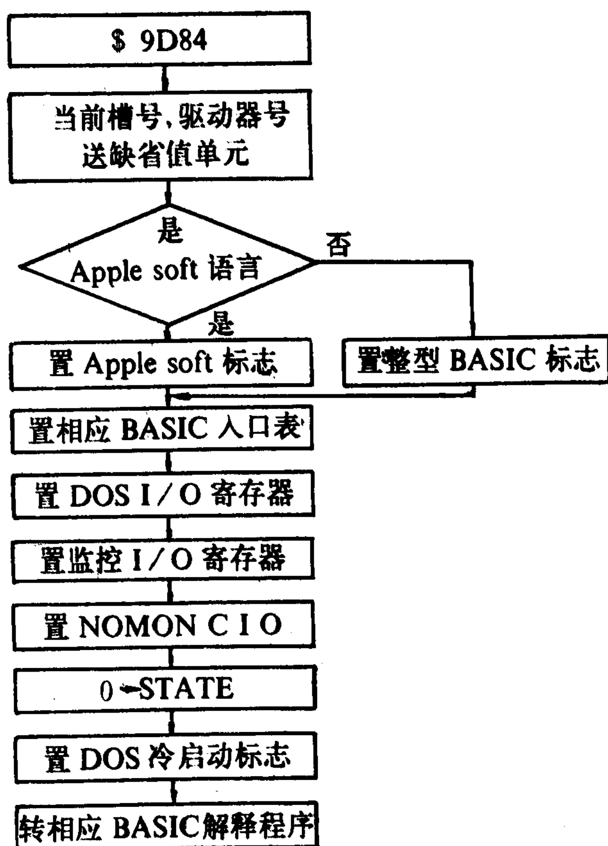


图 5.3.4　冷启动程序框图

3.3.1 DOS输入程序($ 9E81)

该程序的主要工作是:从当前输入设备上取得一字符。若是冷启动,第一次进入 DOS,则执行一段冷启动的初始化工作。图 5.3.5 为 DOS 键盘接收程序框图。

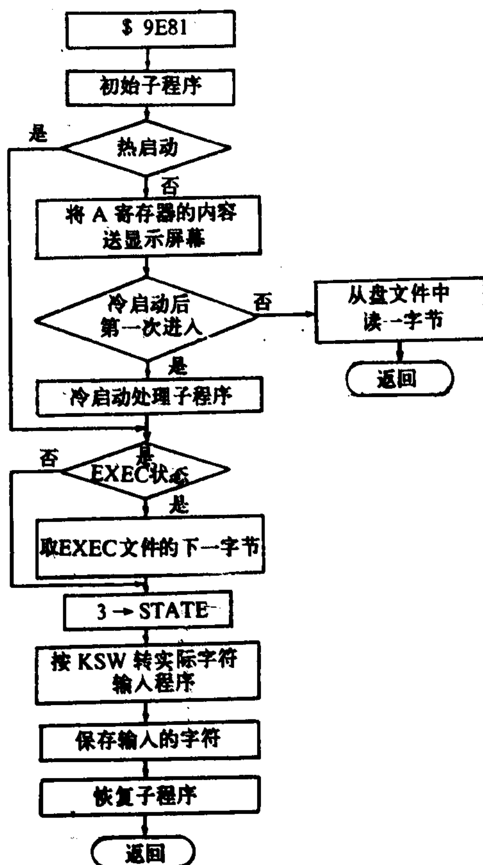


图 5.3.5 DOS 键盘接收程序框图

初始化子程序的工作包括:将 X,Y,S,A 寄存器的值保存到 DOS 的专用单元中,将 DOS 的 I/O 寄存器的值分别放入 KSW 和 CSW 中。

恢复子程序的工作包括:将保存的 X,Y,S,A 寄存器的值恢复,将 CSW 和 KSW 分别置成 DOS 的输出、输入程序的首址。

冷启动处理程序(入口地址: $ 9DEA)主要工作:初始化 BASIC 的标志单元,置 MAXFILES 的缺省值,初始化文件缓冲器,置 STATE 为 0 并置热启动标志,建立第三页向量表,最后执行 HELLO 程序。

3.3.2 DOS 输出程序($ 9EBD)

该程序的主要工作:从当前输出设备上输出一个字符。同时查看输出的内容是否是命令,若是则截取 DOS 命令并转向命令扫描程序去识别。参见图 5.3.6。

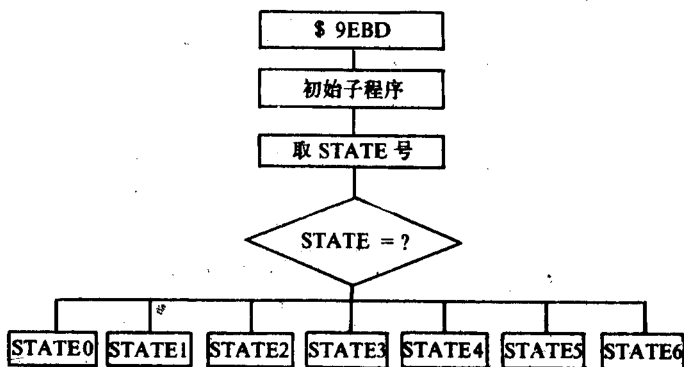


图 5.3.6 屏幕输出入口程序

1．STATE 0：输入行处理程序，从中区分出是否应截取 DOS 命令(图 5.3.7)。

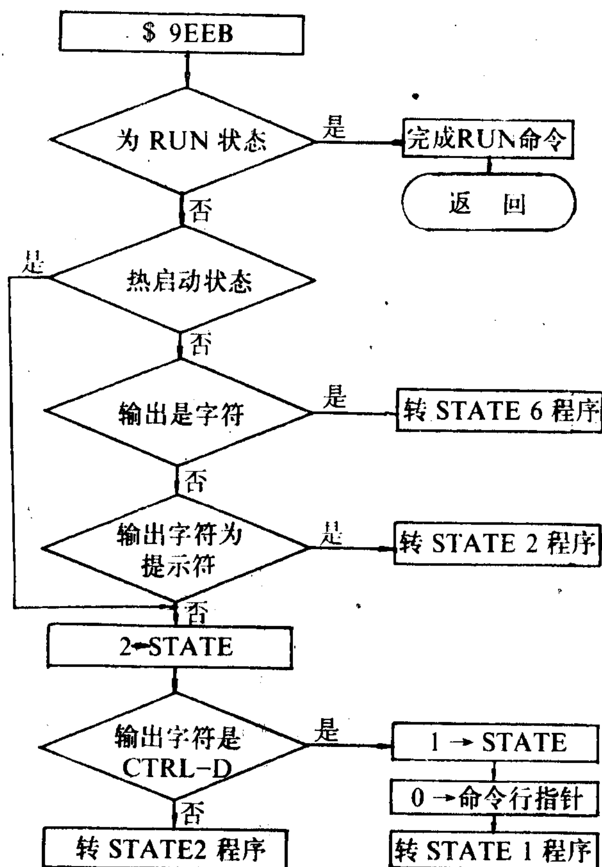


图 5.3.7　STATE 0 程序

2．STATE 1：收集 DOS 命令，并转入命令扫描程序(图 5.3.8)。

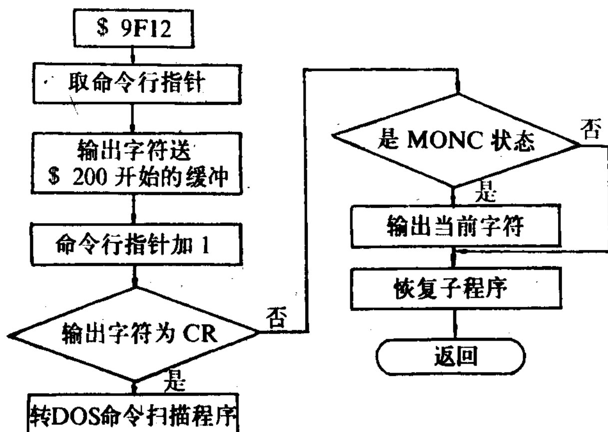


图 5.3.8 STATE 1 程序

3. STATE 2: 完成非 DOS 命令的字符输出(图 5.3.9)。

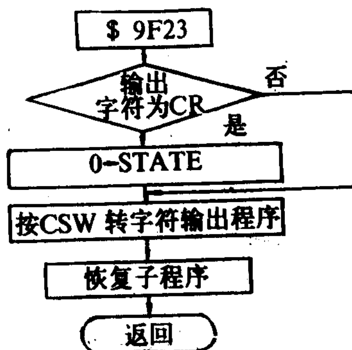


图 5.3.9 STATE 2 程序

4. STATE 3: 完成对 INPUT 命令的处理,若是在 EXEC 状态下,该输出字符串将作为命令执行(图 5.3.10)。

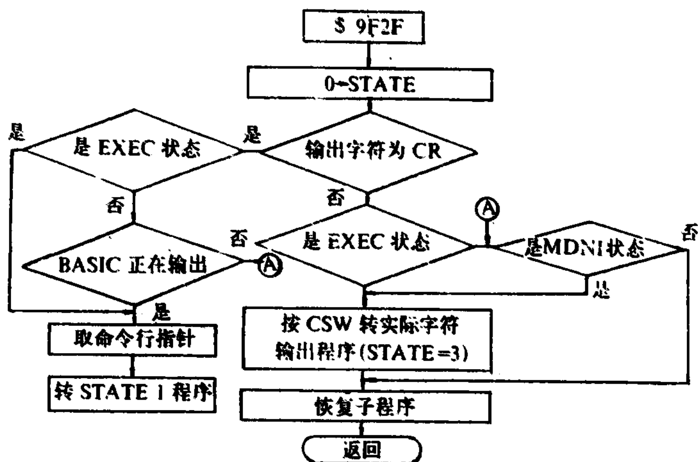


图 5.3.10 STATE 3 程序

5. STATE 4:向文件写数据的处理(图 5.3.11)。

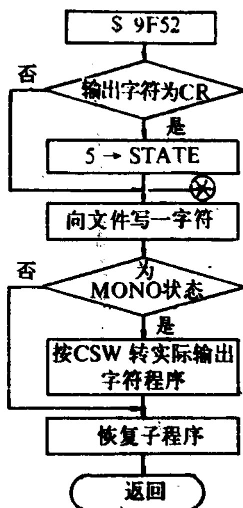


图 5.3.11 STATE 4 程序

6. STATE 5:写入数据行的处理(图 5.3.12)。

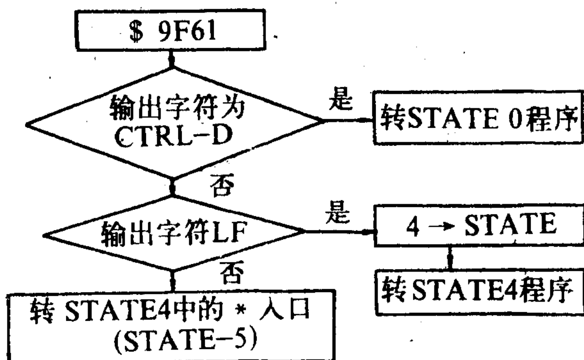


图 5.3.12 STATE 5 程序

7. STATE 6:跳过提示符号(图 5.3.13)。

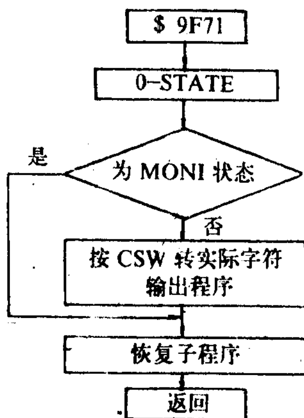


图 5.3.13 STATE 6 程序

3.4 DOS命令扫描程序

由 STATE 1 完成 DOS 命令输入之后;即进入 DOS 命令扫描程序,它将完成对 DOS 命令的判断、识别和参数处理,然后转入各命令处理程序。参见图 5.3.14 和图 5.3.15。

在该程序运行过程中使用了三种表格:

(1) 命令名称表($ A884 ~ $ A908):存放 DOS 命令名称的 ASCII 码。

(2) 命令键字表($ A909 ~ $ A940):存放 DOS 命令应提供的参数标识符,程序将按照该表进行参数的接收处理。

(3) 参数标识符号表($ A941 ~ $ A94A):存放参数标识符的符号。

3.5 DOS命令执行程序

当 DOS 命令扫描程序识别了一个正确的 DOS 命令后,将转入命令执行程序(图 5.3.16)。DOS 命令入口地址如表 5.3.1 所述。

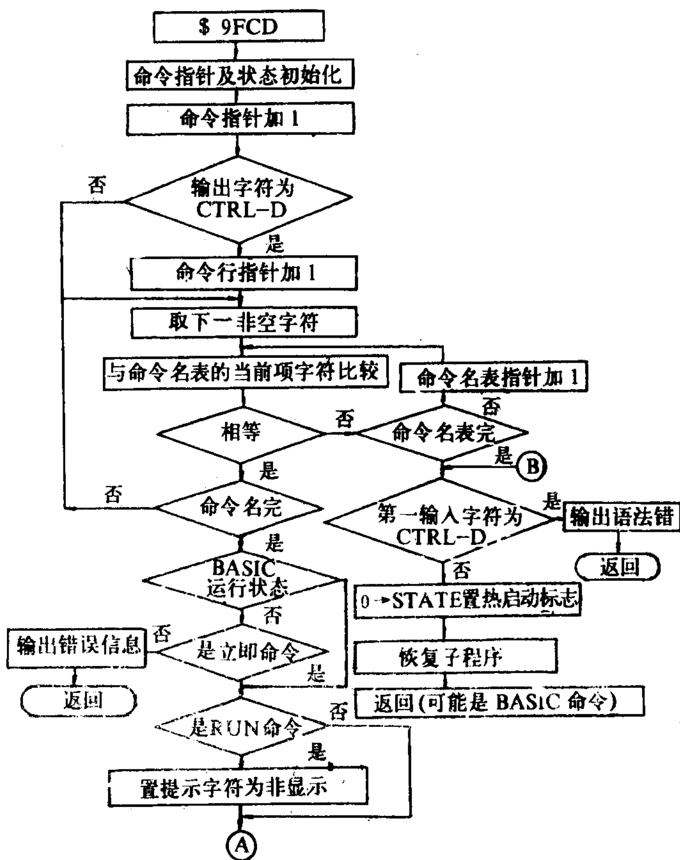


图 5.3.14 DOS 命令扫描程序(一)

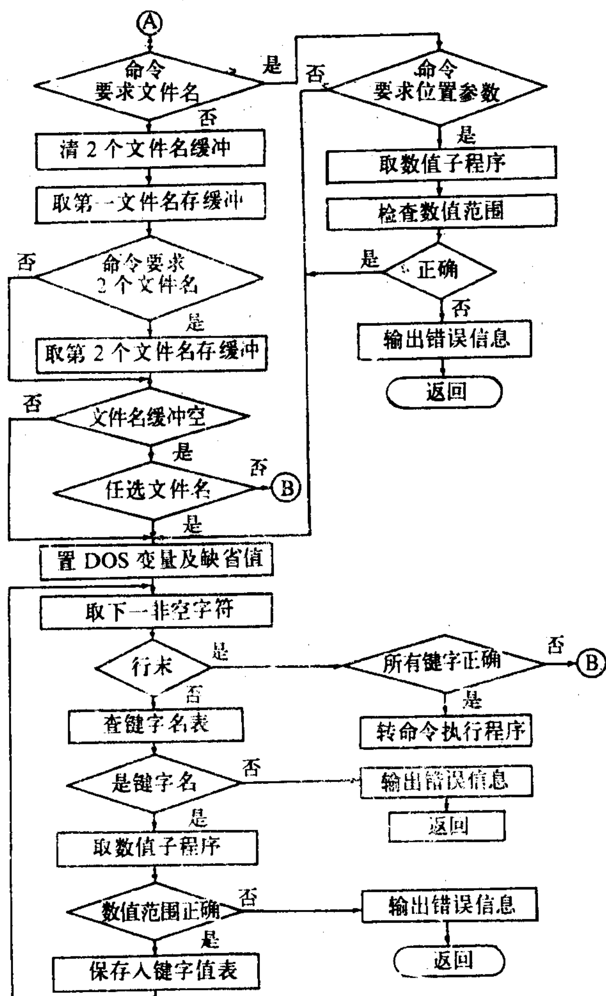


图 5.3.15　DOS 命令扫描程序(二)

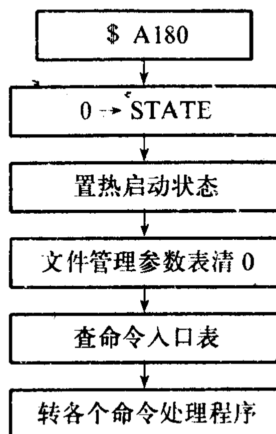


图 5.3.16　DOS 命令执行程序

表 5.3.1　DOS 命令入口地址

命令名称	入口地址	命令名称	入口地址	命令名称	入口地址
INIT	$ A54F	READ	$ A51B	NOMON	$ A23D
LOAD	$ A413	EXEC	$ A5C6	PR #	$ A229
SAVE	$ A397	WRITE	$ A510	IN #	$ A22E
RUN	$ A4D1	POSITION	$ A5DD	MAXFILES	$ A251
CHAIN	$ A4F0	OPEN	$ A2A3	FP	$ A57A
DELETE	$ A262	APPEND	$ A298	INT	$ A59E
LOCK	$ A271	RENAME	$ A281	BSAVE	$ A331
UNLOCK	$ A275	CATALOG	$ A56E	BLOAD	$ A35D
CLOSE	$ A2EA	MON	$ A233	BRUN	$ A38E
VERIFY	$ A27E				

第 四 章

文件管理程序

文件管理程序用来完成对磁盘文件的各种操作和管理，为用户提供一组对磁盘文件进行操作的命令。其中包括有：OPEN，READ，WRITE，LOCK，UNLOCK，RENAME 等 13 条命令。

DOS 文件管理程序是一个接口明确的模块程序，它允许 DOS 系统程序和用户程序来完成对指定文件的各项操作。

4.1 文件缓冲区

文件缓冲区(图 5.4.1)是磁盘文件与主机内存之间进行信息交换的过渡单元，DOS 操作系统为每个活动的文件在主存中都分配一个文件缓冲区，DOS 的文件管理程序将通过文件缓冲区来记录各个文件的当前状态和读写的数据。

在 DOS 系统引导后，系统保留了三个文件缓冲区，每个文件缓冲区占用 595 个字节单元。第一个文件缓冲区的首地址由 $ 9D00 指出，以后的文件缓冲区首地址由前一个文件缓冲区中的指针指出。文件缓冲区的格式如表 5.4.1 所示。

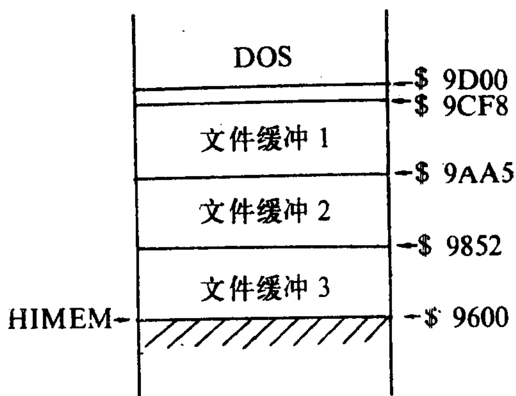


图 5.4.1 文件缓冲区

表 5.4.1 文件缓冲区的格式

字 节	内 容
$ 00 ~ $ FF	数据缓冲区
$ 100 ~ $ 1FF	磁道 / 扇区表区段缓冲区(T / S 表)
$ 200 ~ $ 22C	文件管理工作缓冲区
$ 22D ~ $ 24A	文件名缓冲区(30 个字符)
$ 24B ~ $ 24C	文件管理工作缓冲区的地址
$ 24D ~ $ 24E	磁道 / 扇区表区段缓冲区的地址
$ 24F ~ $ 250	数据缓冲区的地址
$ 251 ~ $ 251	下一文件缓冲区中的文件名缓冲区地址

4.2　文件管理工作区

文件管理工作区又称为 FDB(File Decriptor Block),是用来对指定的文件进行操作控制的,它记录了文件管理的有关信息。

在 DOS 3.3 操作系统中,文件管理工作区在 $ B5D1 ~ $ B5FD 单元。当某一活动文件进行操作时,文件管理程序将从该文件缓冲区中把文件管理工作区缓冲的内容送到文件管理工作区中,待操作执行完毕后,再将相应的信息送回到该文件缓冲区中保存起来。表 5.4.2 列出了文件管理工作区中的有关字节及其所记录信息的作用。

4.3　文件管理程序的调用

一、建立 FMS 表

在调用文件管理程序之前,首先须建立一张"文件管理参数表"(FMS),用以在调用程序和文件管理程序之间交换信息,其格式如表 5.4.3 所示。

表 5.4.2　文件管理工作区字节及其作用

字　节	作　　　用
$ 00 ~ $ 01	本文件 T/S 表的起始位置(T,S)
$ 02 ~ $ 03	当前装入的 T/S 扇区位置
$ 04	标志:80=T/S 区已修改,40=数据区已修改,20=VTOC 已修改,02=上次执行的是写操作
$ 05 ~ $ 06	当前所装的数据区位置
$ 07	文件所在目录扇区的索引
$ 08	文件所在 FDB 的索引
$ 09 ~ $ 0A	一个 T/S 扇区表所含的扇区数,通常为 $ 7A
$ 0B ~ $ 0C	当前所装入的 T/S 扇区的第 5、6 字节单元的值
$ 0D ~ $ 0E	当前 T/S 扇区表所在位置
$ 0F ~ $ 10	与前一次所读扇区的距离
$ 11 ~ $ 12	扇区的字节长度($ FF 字节)
$ 13 ~ $ 14	当前读写位置与文件开头的距离(单位:扇区)
$ 15	当前读写位置与该扇区开头的距离(单位:字节)
$ 16	未使用
$ 17 ~ $ 18	OPEN 时所设定的记录长度
$ 19 ~ $ 1A	当前读写时的记录号
$ 1B ~ $ 1C	当前所读写的记录位置
$ 1D ~ $ 1E	文件长度
$ 1F	下一个要使用的扇区
$ 20	当前正在使用的磁道
$ 21 ~ $ 24	当前正在使用磁道的自由扇区图
$ 25	文件类型及封锁标志
$ 26 ~ $ 27	槽口号 * 16,盘驱动器号
$ 28	磁盘的卷号
$ 29	上次所使用的磁道号
$ 2A ~ $ 2C	未使用

表 5.4.3 FMS 表的格式

字 节	内 容
$ 00	操作命令
$ 01	READ 或 WRITE 命令的子命令
$ 02 ~ $ 09	命令调用的参数说明
$ 0A	回答码
$ 0B	未用
$ 0C ~ $ 0D	文件管理工作缓冲区地址
$ 0E ~ $.0F	磁道 / 扇区表缓冲区地址
$ 10 ~ $ 11	数据缓冲区地址

表中有关内容说明如下:

(1) 操作命令包括:00=空操作, $ 01=OPEN, $ 02=CLOSE, $ 03=READ, 04=WRITE, 05=DELETE, 06=CATALOG, 07=LOCK, 08=UNLOCK, 09=RENAME, 0A=POSITION, 0B=INIT, 0C=VERIFY。

(2) 子命令,仅在 READ 及 WRITE 命令时使用。子命令有:00=空操作,01=R / W 一个字节,02=R / W 一串字节,03=移动到指定位置上 R/ W 一个字节,04=移动到指定位置上 R / W 一串字节。

(3) 参数说明,存放着 R, B, V, D, S 等参数的值,R / W 数据的起始地址、终止地址等。

(4) 回答码,是文件管理程序执行后得到的返回信息。这些信息的意义分别为:00=执行正确,01=LANGUAGE

NOT AVILABLE, 02=操作命令错, 03=子命令错, 04=WRITE PROTECTED, 05=END OF DATA, 06=FILE NOT FOUND, 07=VOLUME MISMATCH, 08=DISK I/O ERROR, 09=DISK FULL, 0A=FILER LOCKED

(5) 最后三个缓冲区地址指的是所访问的文件的文件缓冲区内的相应地址,但也可以安排在用户指定的区域内。

二、将FMS表的首地址放入Y和A寄存器中(低位在前高位在后),并在X寄存器中填入适当的值:X=0表示若文件找不到则建立它,X不等于0表示若文件找不到则不建立。

三、执行JSR $ 3D6即调文件管理程序。

4.4 文件管理程序流程

文件管理程序的入口地址放在$ 3D7和$ 3D8之中,其地址是$ AAFD。

文件管理入口表在$ AAC9 ~ $ AAE4单元。针对文本文件的操作处理给出部分命令的程序流程图,如图5.4.2。

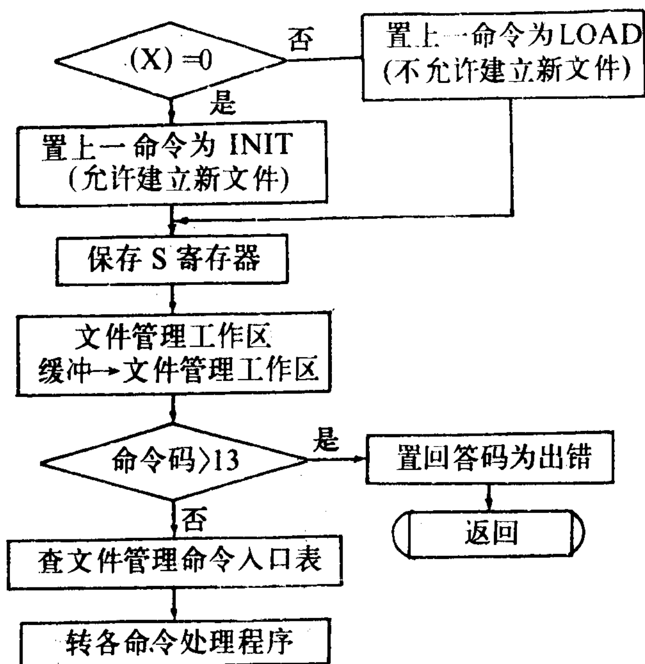


图 5.4.2 文件管理程序流程

第五章
磁盘驱动程序

磁盘驱动程序是DOS的内核程序,它完成磁盘的读写操作,并以磁盘的扇区为单位对磁盘进行读写。该程序也称为"RWTS"程序。

5.1 软盘信息的记录方式和记录格式

5.1.1 道/区的定位

在DOS操作系统下把软盘分为35个读写磁道。实际上磁头从最外道($ 23道)走到最内道($ 00道),步进马达总共步进70个步进道,也就是说可定位70道。由于读/写头的分辨率和步进马达的精度,若使用70道,会产生数据干扰,因而标准DOS 3.3只使用35道。每移动一道,步进马达需前进2个步进道。一般情况下使用的是偶数道。有些系统使用的是其它道,以此作为信息保护的措施。

DOS操作系统对磁盘扇区的定位采用的是"软定位方式"。其方法是:当磁头定位到磁道后,从当前位置开始找出下一地址域,从中读出地址信息,比较是否是要查找的扇区。若不是,再读下一扇区的地址域,直到找到所要的扇区为止。

按照各扇区在磁道上的位置顺序,扇区有逻辑扇区和物理扇区二种概念。物理扇区在盘上是相邻的,而逻辑扇区是不相邻的。调用RWTS时所给的扇区号或用户所使用的扇

区号都是指逻辑扇区号，在 RWTS 程序中将转换成物理扇区号进行操作。逻辑扇区与物理扇区的对应关系为：

物理扇区：0 1 2 3 4 5 6 7 8 9 A B C D E F

逻辑扇区：0 D B 9 7 5 3 1 E C A 8 6 4 2 F

(对于不同的操作系统，逻辑扇区的安排有可能是不同的)。

这是由于大多数磁盘操作是对相邻扇区进行的，在前面一个扇区进行读写处理后，磁头往往已旋转过一定的位置，已越过了相邻的下一个扇区，所以若将逻辑扇区依次相邻编号是不利的。相反，若使各扇区的逻辑位置有一定的间隔，那么从概率角度考虑，可以减少寻区的时间，提高信息的处理速度。

5.1.2 磁盘信息的记录方式

每一个字节有 8 位(bit)，它作为串行数据记录在盘上。各个数据位之间有一个时钟位作为分隔(图 5.5.1)。

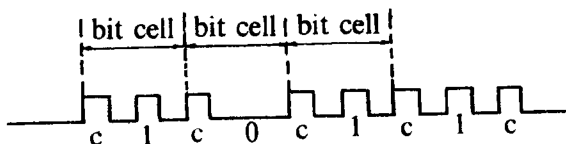


图 5.5.1

图中的 C 表示是时钟位，两个时钟位之间是一个信息位，为高表示是 1，否则为 0。由一个时钟位和一个数据位构成一

个“位单元”(bit cell);每个位单元的时间为:4ms,因此读写一个字节的延迟时间是32ms。

从软件的角度来说,把盘驱动器接口看成是一个数据寄存器。程序中只负责把一个数据字节送到接口,然后由硬接口完成一个字节的写(读)工作。当写盘时,接口将数据寄存器中的字节逐位记入盘中,先高位,后低位,并产生时钟位加以分隔,见图5.5.2。当左移时,不断以0补充进寄存器。若以大于32ms(32周期)来写盘,则将把额外的零记入盘中。

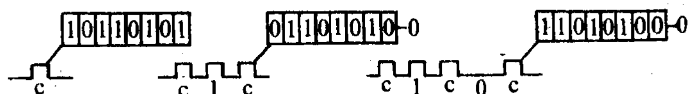


图5.5.2

读出时也是先高位,后低位。由硬件识别时钟位,从而定位数据位,每读入一位,数据寄存器左移一位,同样以32ms读一个字节(图5.5.3)。

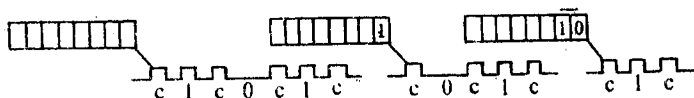


图5.5.3

5.1.3 磁盘信息的记录格式

磁盘的每一个扇区中,除了含有用户的 256 个字节数据外,还有一些标识信息。从这个角度上来看,每一个扇区的信息分为二个域(field),即**地址域和数据域**。

地址域存放的是用作扇区位置标识的数据,它是在软盘的格式化过程中一次写入的,在以后的操作中通过读操作来识别。地址域的格式如图 5.5.4 所示:

起始标志			卷号	道号	区号	校验和	结束标志		
$D5	$AA	$96	2 字节	2 字节	2 字节	2 字节	$DE	$AA	$EB

图 5.5.4　地址域格式

其中,卷号、道号、区号及校验和均采用 4–4 编码,占用 2 个字节。

数据域即用户的 256 个字节的数据,其数据的记录采用的是 6–2 编码,所以 256 个字节的数据,实际占用了 342 个字节的磁盘空间。数据域的记录格式如图 5.5.5 所示:

起始标志			用户数据	校验和	结束标志		
$D5	$AA	$AD	342 字节	1 字节	$DE	$AA	$EB

图 5.5.5　数据域格式

请注意:无论是地址域还是数据域都有域的起始标志和结束标志。这里给出的标志是 DOS 3.3 操作系统所采用的标准字节,但是有一些系统或程序盘片则通过修改这些标志

来达到保护盘片信息不被拷贝的目的。

从整个磁盘来看，某一磁道上的信息记录格式如图5.5.6所示：

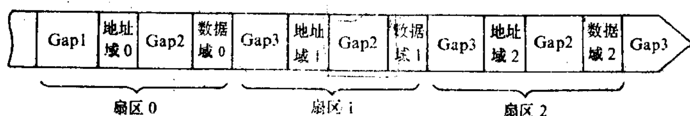


图 5.5.6

域与域之间有一些特殊信息作为间隙(Gap)。间隙由自同步字节组成。自同步字节是一种特殊的字节，它由10bit组成，内容是：1111111100。

间隙的作用是多方面的。首先，它能起字节同步的作用。当磁头定位到道以后，是从该道的任意一个信息位开始读信息，这时所读入的第一位可能不是盘字节的边界，但仍然以8位为一个字节读。当遇到第一个间隙时，由于硬件上要求读入的每个字节的第1位必须为1；所以可验证，磁头通过5个以上的同步字节，必然达到字节同步的目的。例子如图5.5.7所示。



图 5.5.7

间隙还可以保证在地址域读入后，有足够的时间进行地址的译码和判断，而不会使磁头转过了相应的数据域，造成数据丢失。另外，间隙还可防止由于写入前一个域形成的微小误差，影响后一个域的信息，因为机械上的微小误差是不可避免的。

5.2 编码方式

信息是以一定的编码方式记录在软盘上的，为何要对数据进行编码呢？这是由于硬件接口对所识别的信息有一定的限制造成的。首先它不能识别第一位为 0 的数据，也无法识别二个相邻位为 0 的数据等。这样，主机中的十六进制数据就不能直接记录到软盘上，因此就需要采用一定的编码技术，把它们转换一下形式来解决。根据不同的限制条件，可以产生不同的编码方式。

5.2.1 4－4 编码

4–4 编码，是最为简单的一种编码方式，它通过下图的转换，把一个数据字节分成 2 个满足条件的盘字节，记录在盘上。

$$D7\ D6\ D5\ D4\ D3\ D2\ D1\ D0 => \begin{matrix} 1D7\ 1D6\ 1D5\ 1D4 \\ 1D3\ 1D2\ 1D1\ 1D0 \end{matrix}$$

在这种转换中，最高位为 1，且没有二个相邻位为 0，它也就能满足硬件接口识别的要求。这种编码空间利用率低，DOS 3.3 操作系统中，地址域的信息采用的就是 4–4 编码。

5.2.2 5-3编码

在$ 00～$ FF16进制数据中，满足字节最高位为1，且没有二个相邻位为0的字节代码共有34个。例如：AA、AB、AD、AE、AF、B5、...FF。

若取一个字节中的5位，其表示的代码数为32个，用D5、AA作为标志字节，那么剩余的32种代码可以和5位字节建立一一对应的关系：

00 — AB	05 — B6		1B — FA
01 — AD	06 — B7		1C — FB
02 — AE	07 — BA	……	1D — FD
03 — AF	08 — BB		1E — FE
04 — B5	09 — BD		1F — FF

按照这种方法，把一个16进制数据分解成高5位和低3位两部分，低5位部分可用满足要求的盘字节表示，而高3位部分则和后面字节的高3位部分组成另一个满足要求的盘字节，这样便产生了5-3编码。

D7 D6 D5 D4 D3 D2 D1 D0 => D7 D6 D5 D4 D3
D2 D1 D0

采用5-3编码，存储256个字节的数据需要占用410个盘字节的存储空间。在DOS 3.2.1以前版本操作系统中采用了5-3编码方式来存放数据域的信息。

5.2.3 6-2编码

在对Apple软盘驱动器接口电路进行改进后，使得读盘

过程中识别信息的能力有所改善：除了最高位仍必须为1以外，允许仅有一对连读的0存在。

在00～FF代码中，满足这种条件的字节代码共有69个。如果我们取一个字节中的6位信息，它仅需要64种代码即可。如若加上至少有一对相邻的1存在的限制，所需的代码数可变为66个。仍取D5、AA作为标志字节，这样64位字节就能和64个满足要求的代码字节建立一一对应的关系。

00 — 96	05 — 9E		3B — FB
01 — 97	06 — 9F		3C — FC
02 — 9A	07 — A6	……	3D — FD
03 — 9B	08 — A7		3E — FE
04 — 9D	09 — AB		3F — FF

我们取这种对应关系的代码字节作为盘字节。并把一个数据字节分成高6位和低2位二部分。低6位部分可用满足要求的盘字节表示，高2位部分和后面字节中的高2位部分组成另一满足要求的盘字节，这样便产生了6–2编码。

D7 D6 D5 D4 D3 D2 D1 D0 => D7 D6 D5 D4 D3 D2
D1 D0

采用6–2编码方式存储256个字节的数据需要占用342个盘字节空间。显然，它的盘空间利用率要比5–3编码方式有所提高。在DOS 3.3操作系统中采用了6–2编码方式来存放数据域中的信息。

5.3 磁盘驱动程序的调用

磁盘驱动程序(RWTS)是DOS操作系统的最底层程序,它主要用来完成对磁盘的格式化,按磁盘的扇区来读写数据等等。RWTS是作为文件管理程序的一个子程序被调用的。用户程序也可以通过对它的直接调用来快速地读写磁盘数据。

在调用RWTS程序之前,需要建立二张表,用以交换信息。这二张表是I/O控制表(IOB)和设备特征表(DCT),分别如表5.5.1和表5.5.2所示。

按要求建立好这二张表格后,将IOB表的起始地址放在Y和A寄存器中,转入磁盘驱动程序的入口。即执行:JSR $ 3D9。

5.4 磁盘驱动程序流程

磁盘驱动程序的入口地址放在$ 3DA和$ 3DB之中,其地址为$ BD00。图5.5.8为磁盘驱动程序流程图。

5.4.1 数据的编码与解码

从磁盘驱动程序流程图中可以知道,在写数据操作前,先要对数据进行预处理,即进行数据的编码。DOS操作系统中开辟了一个342个字节的数据缓冲区,用来存放满足读写识别用的6位盘字节。这个缓冲区分成二部分,第一缓冲区从$ BB00 ~ $ BBFF,共256个字节,第二缓冲区从$ BC00 ~ $ BC56,共86个字节。

数据预处理子程序的工作是将存放在用户缓冲区的

表 5.5.1 IOB 表

字节	内容
$ 00	表格类型,规定为 $ 01
$ 01 ~ $ 02	槽号 * 16,驱动器号
$ 03	磁盘卷号
$ 04 ~ $ 05	磁道号($ 00 ~ $ 22),扇区号($ 00 ~ $ 0F)
$ 06 ~ $ 07	设备特征表(DCT)的地址
$ 08 ~ $ 09	256 个字节的读写缓冲区的首地址
$ 0A ~ $ 0B	未用
$ 0C	命令码, 00 = 查找(SEEK), 01 = 读扇区(READ), 02 = 写扇区(WRITE), 04 = 格式化(FORMAT)
$ 0D	回答码, 00 = 无错误, 01 = 写保护, 02 = 卷号不匹配, 40 = I / O 错, 80 = 读错
$ 0E	上次访问时磁盘的卷号
$ 0F ~ $ 10	上次访问时, 盘驱所在槽号 * 16, 驱动器号

表 5.5.2 DCT 表

字节	内容
$ 00	设备类型,规定为 $ 00
$ 01	每道的步进道数,规定值为 $ 01
$ 02 ~ $ 03	马达启动后稳定时间,规定值为 $ EFD8

256个字节，经编码后组成342字节的6位盘字节数据，存放在DOS系统的第一和第二缓冲区中。数据的组织方法参见图5.5.9。

数据预处理程序在＄B800～＄B829，其程序流程如图5.5.10所示。

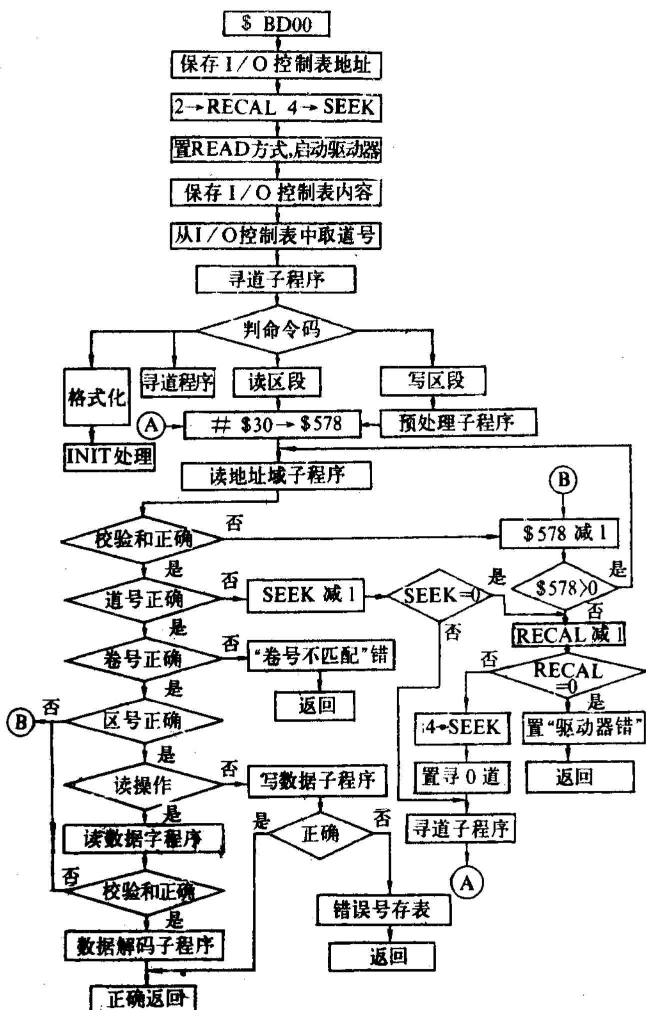
在读磁盘扇区操作中，当数据读入后，要对它进行解码操作，它是编码操作的逆过程，其目的是把读入的盘字节转换成16进制的机器码。

数据解码程序在＄B8C2～＄B8DB。其程序流程如图5.5.11所示。

5.4.2 写数据子程序

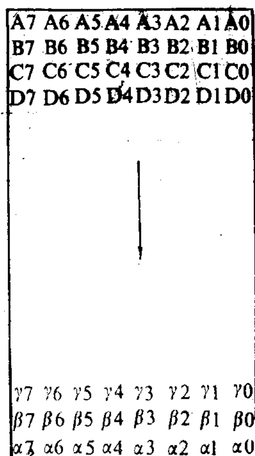
用户缓冲区的数据经数据编码子程序预处理之后，可由写数据子程序完成写盘操作。在逐个字节写盘的同时，程序要完成二项工作：一是计算校验和，二是将6位数据转换成相应的盘字节。

转换成盘字节的工作是通过“写数据转换表”来完成的，该表在主存的＄BA29～＄BA69单元。其转换过程可由图5.5.12来说明。



$ BD00
保存 I/O 控制表地址
2→RECAL 4→SEEK
置READ方式,启动驱动器
保存 I/O 控制表内容
从I/O控制表中取道号
寻道子程序
判命令码
格式化
寻道程序
读区段
写区段
INIT处理
A
$30→$578
预处理子程序
读地址域子程序
B
否
校验和正确
$578 减 1
是
是
否
道号正确
SEEK 减 1
SEEK=0
$578>0
否
是
否
RECAL 减 1
卷号正确
否
"卷号不匹配"错
是
否
RECAL =0
返回
是
B
否
区号正确
4→SEEK
置"驱动器错"
是
置寻 0 道
返回
否
读操作
写数据子程序
是
是
否
读数据字程序
正确
寻道子程序
否
校验和正确
A
是
错误号存表
数据解码子程序
返回
正确返回

图 5.5.8 磁盘驱动程序



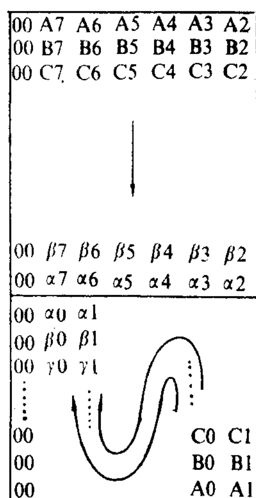


图 5.5.9 数据的组织方法

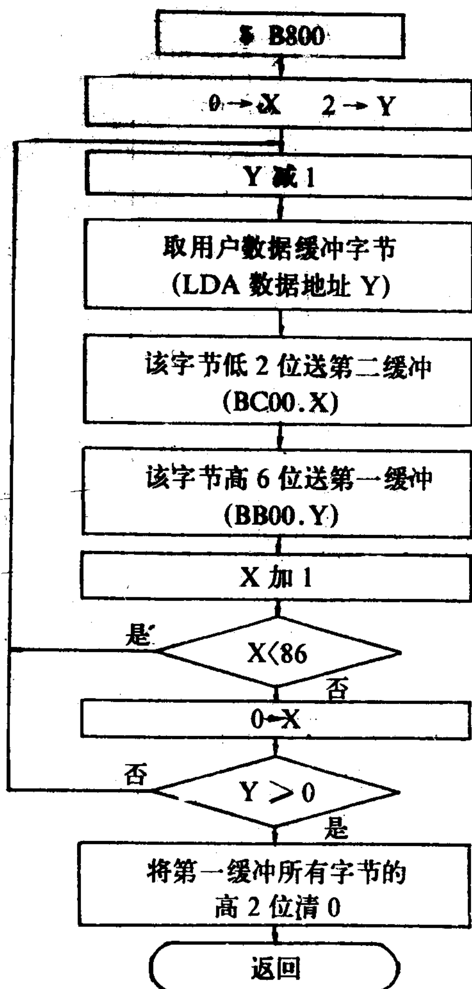


图 5.5.10 数据编码程序

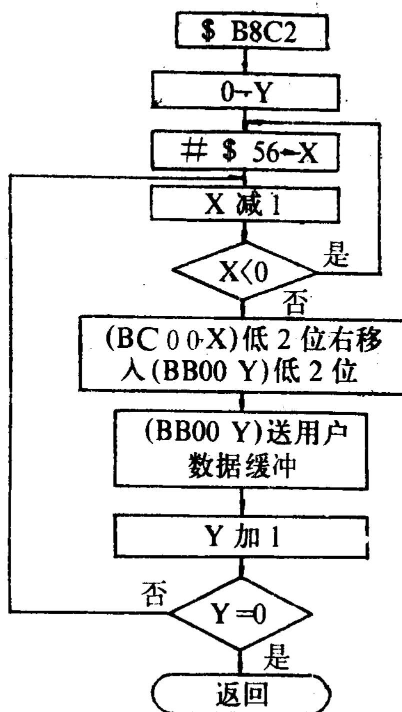
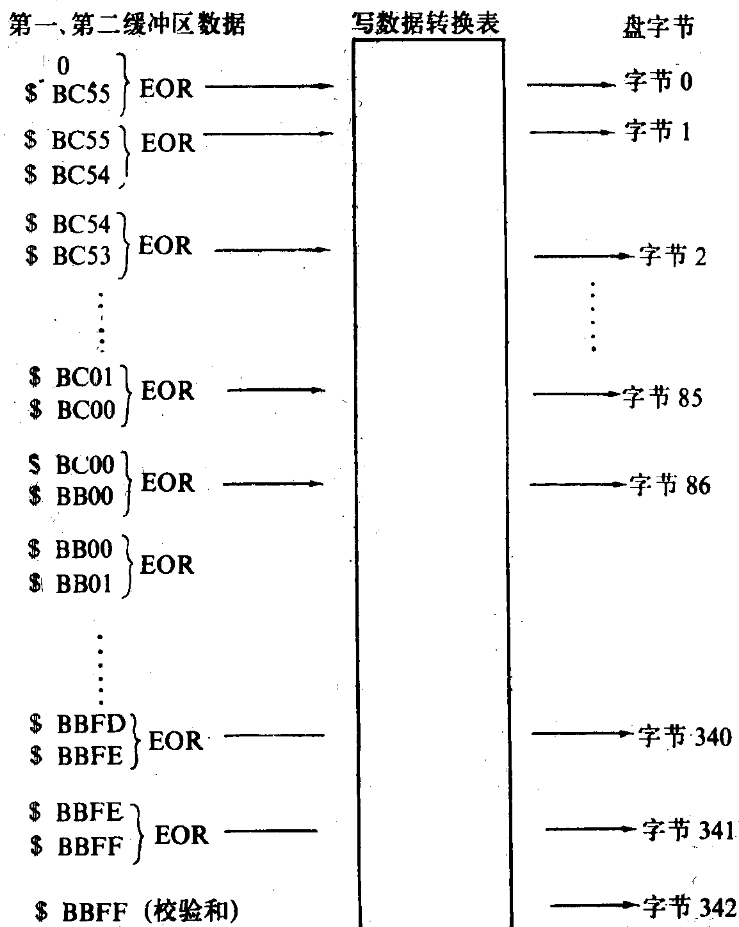


图 5.5.11 数据解码程序



第一、第二缓冲区数据
写数据转换表
盘字节
0
$ BC55
EOR
字节 0
$ BC55
$ BC54
EOR
字节 1
$ BC54
$ BC53
EOR
字节 2
$ BC01
$ BC00
EOR
字节 85
$ BC00
$ BB00
EOR
字节 86
$ BB00
$ BB01
EOR
$ BBFD
$ BBFE
EOR
字节 340
$ BBFE
$ BBFF
EOR
字节 341
$ BBFF (校验和)
字节 342

图 5.5.12 写数据转换

写数据子程序在 $ B82A ~ $ B8B7。其程序流程如图 5.5.13 所示。

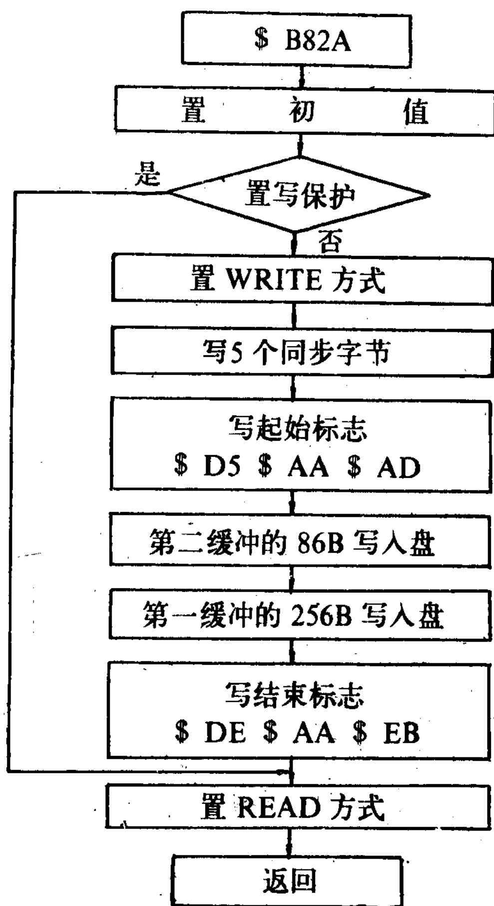


图 5.5.13 写数据子程序流程图

5.4.3 读数据子程序

读数据子程序是写数据的逆过程，它将每个字节从盘上读出后，将通过“读数据转换表”把盘字节转换成 6 位字节，存放到第一和第二缓冲区内。最后再由数据解码程序完成 6 位字节到 8 位字节的转换工作，并存入到用户数据缓冲区中。

“读数据转换表”在内存的 $ BA96 ~ $ BAFF 单元。其转换过程可由图 5.5.14 来说明。

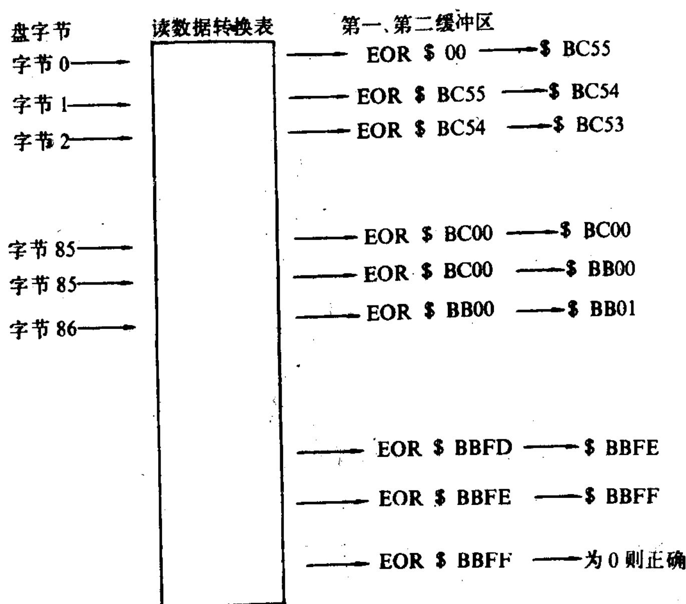


图 5.5.14 读数据转换

读数据子程序在＄B8DC～＄B944。其程序执行流程如图 5.5.15 所示。

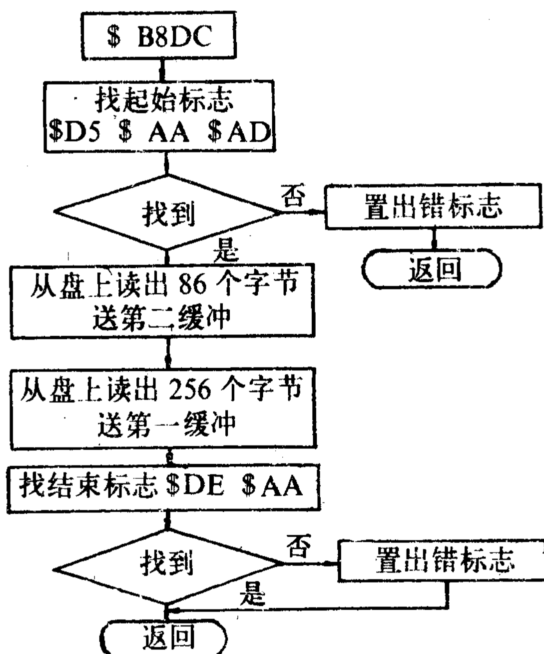


图 5.5.15　读数据子程序流程图

经过上面对磁盘驱动程序的分析，可以将该程序的处理过程归纳成写数据、读数据二个过程。其处理流程如图 5.5.16 所示。

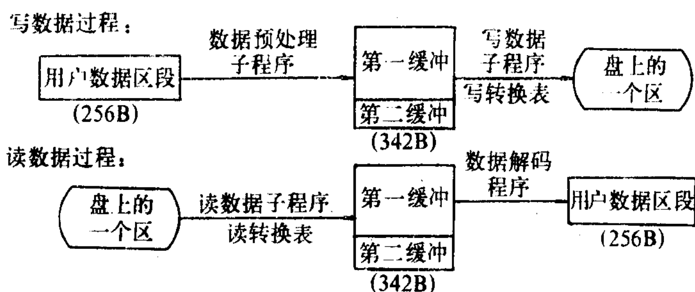


图 5.5.16　数据处理流程图

附录 A DOS 操作系统的内存分配图

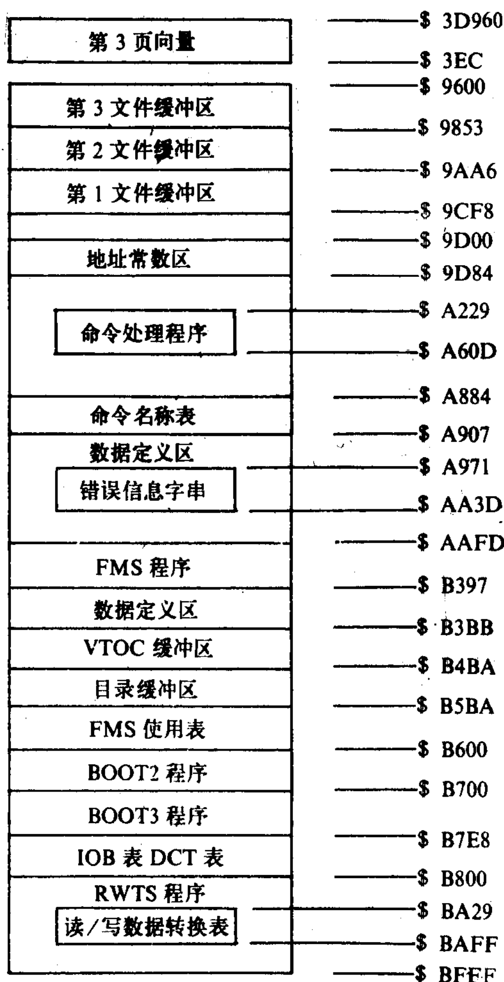


图 5.A.1 DOS 操作系统的内存分配图

第三篇　CEC—2000 硬件系统简介

第一章　存储器系统

微处理器 6502 可寻址 64KB 存储空间。CEC—2000 型学习机中 ROM、RAM、以及输入输出设备所占用的存储空间，都分布在这 64KB 地址内。在学习机内，所用存储空间大于 64KB。因此，采用了存储体的切换技术，使同一逻辑地址空间可以映射到多个同样大小的物理地址空间，通过设置存储体切换控制开关，控制逻辑地址到物理地址的映射，使得某一逻辑地址空间可以分时映射到多个同样大小的物理地址空间。这样，大大地扩展了学习机的存储器容量。

CEC—2000 中华学习机存储空间可以分为主存储器空间及辅存储器空间两大部分。主存储器空间是系统工作时使用的最基本的空间，包括有主 RAM、ROM 及 I/O 空间。辅存储器空间则包括汉字系统所占用的空间及扩充 RAM 占用空间。

· 主存储器空间

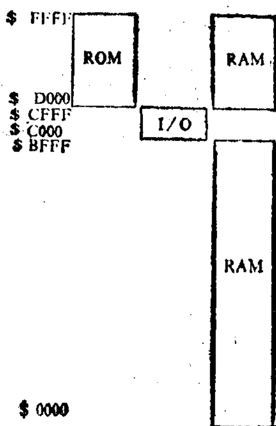
CEC—2000 型学习机的主存储器分配如图 1.1 所示。

空间 $ 0000— $ BFFF 为主 RAM 空间。输入输出空间在 $ C000— $ CFFF。空间 $ D000— $ FFFF 为存储体切换空间，此空间可以映射成 ROM，也可以映射成 RAM。

1.1 读/写存储器(RAM)

一、RAM 存储空间分配

前面已经提到主存储器空间有 64K 字节的 RAM。这 64K 字节的空间区分成两大部分，主 RAM 的基本部分在空间 $0000－$BFFF，此空间总共有 48K 字节。另一块主 RAM 为存储体切换空间，地址范围为 $D000－$FFFF。这里，首先介绍一下主 RAM 的基本部分。



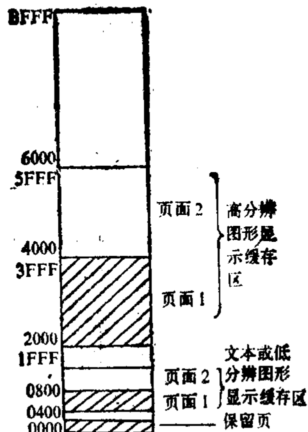


图 1.1 主存储器分配图　　　　图 1.2 主 RAM 基本部分分配图

主 RAM 基本部分的分配见图 1.2，在这地址空间中，包括系统使用的保留页面和显示缓存区。

· 保留页面

主 RAM 的绝大部分都提供给用户存放程序和数据。但还有少量页面作为系统工作区，供监控程序、BASIC 解释程序使用。

系统并不能阻止用户使用系统工作区。若用户要访问系统工作区时，应特别注意，不要破坏系统的数据，以免导致系统的错误动作。

· 零页

在 6502 的寻址方式中，有几种方式需要使用到零页空间 $ 0000－$ 00FF。使用到零页的寻址方式都具有指令短、寻址灵活的特点。监控程序、BASIC 解释程序以及 DOS 都大量地使用零页，以提高程序的效率。

· 系统堆栈

6502 使用第 1 页（$ 0100－$ 01FF）作为堆栈，在此空间，6502 按照先进后出的原则，存取数据，由堆栈指针指示堆栈中当前可访问的单元。

· 输入缓存区

输入子程序 GETIN 使用第 2 页（$ 0200－$ 02FF）作为键盘输入缓存区。监控程序和 BASIC 都使用 GETIN。这个输入缓存区的大小决定输入字符串的最大字符数。一般允许输入 256 个字符，但只取前 237 个。假若可预知将不输入任意长的字符串，就可以使用第 2 页的高地址单元作为数据暂存单元。

· 显示缓存区

在文本显示方式和低分辨图形方式中，显示第一页在空间 $ 0400－$ 07FF。这 1024 个字节不能作为程序及数据的存放区。在这区域里，有 64 个存储单元的内容不在屏幕上显示，这些单元留给外围扩展卡使用。

文本方式和低分辨图形方式的显示第二页，占用了空间 $ 0800－$ 0BFF。也是 1024 个字节。大部公程序都不使用此

页作为显示缓存区。可以根据情况，将该空间作为程序和数据的存放区。

高分辨图形显示缓存区的第一页占用空间＄2000－＄3FFF。如果不用高分辨图形显示时，该空间可以存放程序和数据。

高分辨图形显示缓存区的第二页占用空间＄4000－＄5FFF。同样，高分辨图形显示不用此页显示时，它可以存放程序和数据。

关于显示缓存区的详细情况参见视频显示部分。

二、存储体切换空间

地址空间＄D000－＄FFFF称为存储体切换空间。它可以映射到两个物理空间，一是12K ROM，用来存放监控程序和BASIC解释程序，另一物理空间为16K RAM，一般有来存放整数BASIC或其它应用程序。

在这里12K的地址空间映射到16K RAM空间，这是因为地址空间＄D000－＄DFFF可以映射到两个4K RAM，即访问16K RAM时，要对＄D000－＄DFFF空间访问两次，分别访问两个4K RAM存储空间。



图1.3存储体切换空间分配图

切换不同的存储模块被映射到同一地址空间称为存储体的切换。在空间＄D000－＄FFFF的存储体切换有两层含意：

一是在这 12K 地址空间进行 ROM 和 RAM 之间的切换；二是在地址 $D000－$DFFF 这 4K 空间进行到两个 4K RAM 的切换。这一地址空间的分布见图 1.3。

存储体的切换是通过改变软开关的状态来实现的。在此空间进行存储体切换，要通过设置软开关的状态来做三个选择：一是该地址空间映射到 ROM 还是 RAM；二是允许写 RAM 还是禁止写 RAM(写保护)；三是地址 $D000－$DFFF 是映射到第一个 4K RAM 还是映射到第二个 4K RAM。

表 1.1 存储体空间切换软开关地址

软开关	写	读	读	4K	RAM 体
地　址	RAM	RAM	ROM	第一体	第二体
$C080		•			•
$C081	•		•		•
$C082			•		•
$C083	•	•			•
$C084		•			•
$C085	•		•		•
$C086			•		•
$C087	•	•			•
$C088		•		•	
$C089	•		•	•	
$C08A			•	•	
$C08B	•	•		•	
$C08C		•		•	
$C08D	•			•	
$C08E			•	•	
$C08F	•	•		•	

表 1.1 列出设置空间 $D000－$FFFF 存储体切换的软开关地址。通过对这些地址的访问，可设置相应的软开关，实现对该空间的存储器切换。这些地址在 $C080－$C08F。

1.2 汉字所用存储空间

CEC－2000 学习机具有汉字处理功能。它采用主板的微处理器完成汉字处理全过程，包括实现汉字输入码到内码的转换，汉字字形点阵的查找，屏幕显示编辑以及汉字打印输出。CEC－2000 学习机配备一、二级国际汉字，汉字字形显示为 16×16 点阵。由于汉字的字形采用全点阵，字形库容量为 2 兆位，即 256K 字节。在汉字工作方式中，汉字处理程序，以及实现输入码到内码转换的转换码表也都需要相当的存储容量。要实现主 CPU 完成汉字处理，就必须扩充主 CPU 可访问的存储空间，提供实现汉字处理的基本环境。

学习机所用的微处理器 6502 可寻址 64K。在 MMU 的管理下，6502 可访问主存储器中 16K ROM，64K RAM。另外，利用设置 MMU 内部软开关，还可访问和 RAM 相对应的辅存空间。其地址为 $0000－$BFFF 以及 $D000－$FFFF。用于主存储器和辅存储器之间切换的软开关有：

RAMRD；RAMWRT；ALTZP。

关于这些软开关的功能在 MMU 部分进行介绍。

在 CEC－2000 学习机中，辅存空间用做汉字处理所需存储器的扩充。辅存空间分成 4 块。3 块为 ROM 区，1 块为 RAM 区。辅存分配如图 1.4。

第一块地址空 $0000－$3FFF

这 16K 空间是 RAM 区。在汉字工作方式中，它作为 RAM 工作区。对于这一块 RAM 区，有一点需要说明，它并没

有映射到一个单独 RAM 存储器。它所映射的 RAM 存储器就是主 RAM 里同一地址所映射的 RAM 存储器。这样，在辅存中访问的 16K RAM 也就是主存中同一地址的 RAM。

第二块地址空间 $4000— $7FFF。

这 16K 空间是 ROM 区。用于访问汉字点阵库。为实现对 256K 点阵的访问，利用这 16K 空间以及存储体切换电路。切换电路提供 4 位扩展地址线。对汉字点阵库的 16 个存储体切换选择。从而实现对 256K(16×16K)点阵库空间访问。存储体的切换，将在后面介绍。

第三块地址空间 $8000— $BFFF。

这 16K 是 ROM 区。用于存放转换码表。在汉字输入时，利用该表，将输入码转换成内部码。考虑到汉字输入方式的扩充以及系统软件固化，该存储空间有两个存储体，在拼音输入方式中，使用第一体存放转换表。存储体切换，将在后面介绍。

第四块地址空间 $D000— $FFFF。

这 12K ROM 区用于存放汉字处理程序。

1.3 辅存储器的切换

要进行汉字处理，必须申请所需要的存储空间。这样，就需要对存储器进行切换，将当前访问的存储空间切换成辅存储器，提供辅存供汉字处理作用。

从图 1.4 可以看出，在地址上，辅存分为两块，一块地址空间为 $0000— $BFFF；另一块为 $D0000— $FFFF。但实际上，在进行主存和辅存切换时，是将存储空间分成两部分进行的。第一部分为地址空间 $0200— $BFFF，第二部分为 $0000— $01FF 和 $D00— $FFFF。在辅存中，空间 $0000

—$3FFF 作为 RAM 区。此空间映射到主 RAM 所对应的存储器上。这样，不管是在主存对此空间访问，还是在辅存对此空间访问，均访问同一存储器。在进行主存和辅存切换时，对于此空间的使用没有任何影响。

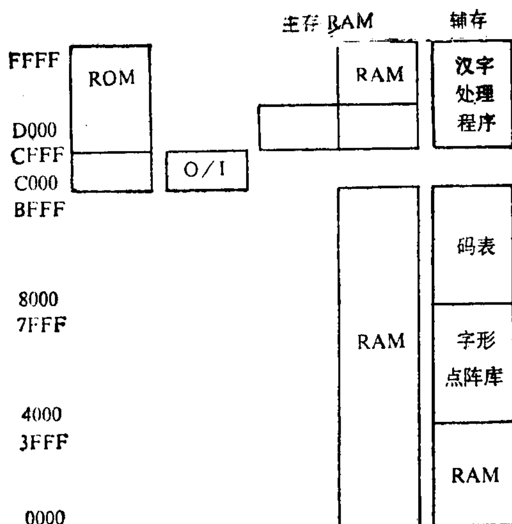


图 1.4 配上汉字处理电路的系统地址分配图

在辅存中，除了空间 $0000—$3FFF 以外，其余均为 ROM 区。由于主存/辅存的切换，并不影响对地址为 $0000—$3FFF 的 RAM 区的使用。在切换存储空间为辅存时，仅需要在辅存中进行读操作，就可以使用辅存的 ROM 区和 RAM 区。对于空间 $0200—$BFFF，仅改变软开关 RAMRD 状态，就可以确定对该空间的读操作是在主 RAM 还是在辅存进行。要将该空间的读操作切到辅存进行时，将 RAMRD 开关状态置为“1”。即执行指令

STA　＄C003

这样，就可以访问辅存中 16K RAM 区，字形点阵库 ROM 区，以及码表 ROM 区。

要将此空间切回主 RAM 区，将软开关 RAMRD 状态置"0"，即执行指令：

STA　＄C002

就可对主 RAM 区＄0200—＄BFFF 进行读写访问。

第二部分空间＄D000—＄FFFF 以及＄0000—＄01FF，在主存和辅存之间切换，由软开关 ALTZP 控制。把空间＄D000—＄FFFF 称为存储体切换空间。在主存中，该空间为 ROM 时，即使进行主存到辅存的切换，即设置 ALTZP 为"1"，此空间也不能切换成辅存。只有在该空间映射为 RAM，如主存该空间允许读 RAM，禁止写 RAM，再置 ALTZP 为"1"就可以将该空间节换成辅存 RAM。关于该空间在主存中进行 ROM 和 RAM 之间的切换，请见"存储体切换空间"。

当空间＄D000—＄FFFF 为主存 ROM 时，执行如下指令：

LDA　＄C080；禁止主 ROM，仅允许读主 RAM。

STA　＄C009；由主存切换到辅存。

就实现从主 ROM 到辅 ROM 的切换。

若要从辅存 ROM 切换到主 ROM，执行如下两条指令即可。

STA　＄C008；由辅存切换到主存。

LDA　＄C082；禁止主 RAM，允许读主 ROM。

在主存和辅存切换时，切换操作指令既不能放在主存中，也不能放在辅存中，否则切换后将会造成混乱和错误。由于主

/辅存切换不影响 I/O 空间，该空间可发和主存在一起使用，也可以和辅存一起使用。因此，选择 I/O 空间作为切换指令执行区。在 CEC—2000 中，选择 $C300— $C3FF 这一页作为汉字处理的入口，以及切换指令执行区。在进入汉字工作方式时，由 I/O 空间的 $C300 单元作为入口，就可以正确进行汉字处理。

在 MMU 中，考虑系统复位后，空间 $C300— $C3FF 映射到内部 ROM，而用于汉字处理的该空间是在输入输出空间。在转向 $C300 单元执行以前，必须确保该空间在输入输出空间，因而要将软开关 SLOT3ROM 的状态置为"1"，SLOTCXROM 状态置为"0"，即执行指令：

```
STA     $C00B
STA     $C006
```

这样，空间 $C300— $C3FF 就映射在输入输出空间。

1.4 字库 ROM 寻址方法

在汉字处理电路中，为扩大存储器容量，采用了存储体切换技术。存储体的切换，是利用对 $C0B0 单元置值来实现的。在访问 $C0B0 时，$D_0$～$D_3$ 作为对空间 $4000— $7FFF 的选体控制。$D_4$ 作为彩色禁止位。D_5 作为选择哪个辅存体(ROM)的控制。具体说明见表 1.2

在汉字工作方式中，采用拼音输入，转换码表放在空间 $8000— $BFFF 的第一体，在给单元 $C0B0 送控制字时，D5 为"1"，即控制字为 001×××××。若选择码表第二体，则 D5 为"0"，即控制字为 000×××××。

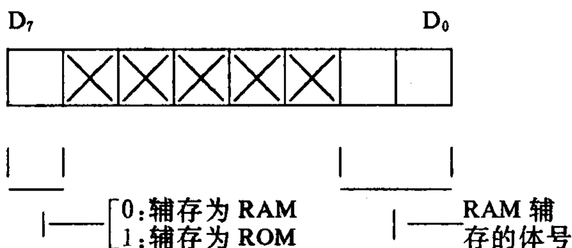
表 1.2 单元 $ C0B0 各位的功能说明

数	据							功　　能
D7	D6	D5	D4	D3	D2	D1	D0	
×	×	×	×	0	0	0	0	选择空间 $ 4000— $ 7FFF 的第 1 体
×	×	×	×	0	0	0	1	选择空间 $ 4000— $ 7FFF 的第 2 体
×	×	×	×	0	0	1	0	选择空间 $ 4000— $ 7FFF 的第 3 体
×	×	×	×	0	0	1	1	选择空间 $ 4000— $ 7FFF 的第 4 体
×	×	×	×	0	1	0	0	选择空间 $ 4000— $ 7FFF 的第 5 体
×	×	×	×	0	1	0	1	选择空间 $ 4000— $ 7FFF 的第 6 体
×	×	×	×	0	1	1	0	选择空间 $ 4000— $ 7FFF 的第 7 体
×	×	×	×	0	1	1	1	选择空间 $ 4000— $ 7FFF 的第 8 体
×	×	×	×	1	0	0	0	选择空间 $ 4000— $ 7FFF 的第 9 体
×	×	×	×	1	0	0	1	选择空间 $ 4000— $ 7FFF 的第 10 体
×	×	×	×	1	0	1	0	选择空间 $ 4000— $ 7FFF 的第 11 体
×	×	×	×	1	0	1	1	选择空间 $ 4000— $ 7FFF 的第 12 体
×	×	×	×	1	1	0	0	选择空间 $ 4000— $ 7FFF 的第 13 体
×	×	×	×	1	1	0	1	选择空间 $ 4000— $ 7FFF 的第 14 体
×	×	×	×	1	1	1	0	选择空间 $ 4000— $ 7FFF 的第 15 体
×	×	×	×	1	1	1	1	选择空间 $ 4000— $ 7FFF 的第 16 体
×	×	×	0	×	×	×	×	禁止彩色图形显示
×	×	×	1	×	×	×	×	允许彩色图形显示
×	×	0	×	×	×	×	×	选择空间 $ 8000— $ BFFF 的第 1 体
×	×	1	×	×	×	×	×	选择空间 $ 8000— $ BFFF 的第 2 体

1.5　320KRAM 扩充系统

在 CEC—2000 主机板上，有一组插座是用于扩充 RAM 的。插上 CEC—2000 专用 RAM 扩充卡，就可以再扩充 256KRAM 存储器(再大容量的扩充卡可定做)，加上主板上原有的 64KRAM，就能使 CEC—2000 达到 320KRAM 的存储空间。扩充 RAM 系统后，不仅增加了内存，还可使显示方式增加分辩率。

扩充的 RAM 地址也挂在辅存空间，地址范围为 0～$FFFF，见图 1.5。由于辅存上还挂有 ROM 存储器(汉字管理系统)，因此，必须把辅存 RAM 和辅存 ROM 分开。另外，由于 6502CPU 的寻址范围为 64KB，所以还必须将 256KRAM 分为 4 个 64K 大小的存储体。为此，CEC—2000 单独设计了一个软切体单元：$C071 去完成上述功能。$C071 各位的功能如下：



在机器复位过程中，系统自动置为 ROM 辅存。如果要兼容 APPLE Ⅱe 工作方式，则可在开机时立即按下 ESC 键，则系统将辅存切至辅存 RAM。请见“第 2 章 2.2 倍高分辨率图形方式”中的内容。除 $C071 这一特殊切换单元外，所有主、辅存体的切换方法详见本章 1.3 节辅存储器的切换。

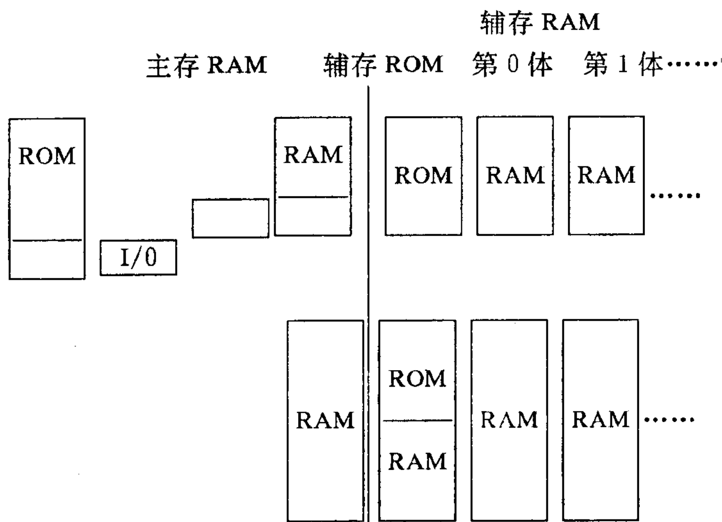


图 1.5 插有 RAM 扩充卡的存储器系统示意图

第二章　显示系统

2.1　显示方式

中华学习机共有 5 种显示方式，主要分两大类，一类为字符(文本方式)，一类为图形方式。图形方式又分低分辨图形和高分辨图形。中文方式采用高分辨显示方式，并通过软件控制屏幕管理。混合方式是字符和图形同时在屏幕上显示，无论在低分辨图形方式或高分辨图形方式下，屏幕下方都留有四行文本的显示区。

一、文本显示

显示文本包括：大小写英文字母，10 个数字、标点符号及其它字符共 96 种字符。

显示格式:全屏幕 24 行,每行 40 个字符,共计 960 个字符。字模 5(宽)×7(高),每个字占据点阵空间 7(宽)×8(高)。

显示方式:(1)正常方式——黑底白字;(2)反向方式——白底黑字;(3)闪烁方式——正常方式和反向方式交替。

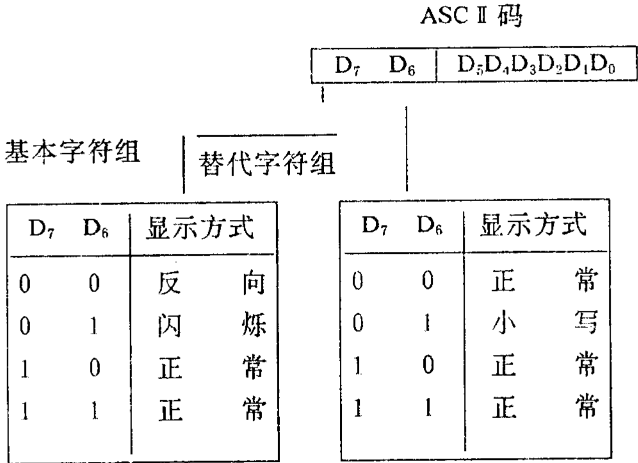
中华学习机可以显示两种文本字符组,一种是基本字符组,另一种是替代字符组,两种字符组外形相同(指点阵模型尺寸相同)而显示的方式不同,这两种字符组的选择由软开关(ALTCHRSET)决定。

显示基本字符组时,大写字母可用上述三种显示,但小写字母仅能用正常方式。显示替代字符组时,没有闪烁方式,但无论大小写字母,数字和特殊字都可以用正常方式和反向方式显示如表:2.1

表 2.1 基本字符组和替代字符组的十六进制码及对应的显示方式

十六进制值	基本字符组		替代字符组	
	字型	方式	字型	方式
$00~$1F	大写字母	反向	大写字母	反向
$20~$3F	特殊字符	反向	特殊字符	反向
$40~$5F	大写字母	闪烁	大写字母	反向
$60~$7F	特殊字符	闪烁	小写字母	反向
$80~$9F	大写字母	正常	大写字母	正常
$A0~$BF	特殊字符	正常	特殊字符	正常
$C0~$DF	大写字母	正常	大写字母	正常
$E0~$FF	小写字母	正常	小写字母	正常

每一个显示在字符的存储器中占一个字节，字节的低 6 位 D5—D0 表示显示字符的 ASCⅠ码，高两位 D7，D6 是用来决定文本的显示方式和大小写字母的选择。在基本字符组中第 7 位决定是正常方式或其它方式，第 6 位决定是否闪烁。在替代字符组中，第 7 位作用与基本字符组相同，第 6 位选择大写或小写字母，参见图 2.1



基本字符组：

D_7	D_6	显示方式
0	0	反向
0	1	闪烁
1	0	正常
1	1	正常

替代字符组：

D_7	D_6	显示方式
0	0	正常
0	1	小写
1	0	正常
1	1	正常

图 2.1 文本方式下 D7D6 选择示意图

二、低分辨率图形方式

低分辨率图形方式和高分辨率图形方式都把屏幕当作矩形点阵来处理。通常使用高级语言在屏幕上画点、线、图形，这里主要说明产生图形的数据在存储器中的存储方式。

低分辨图形方式显示 40×48 的色块阵列，每块可显示包括黑白两色在内的 16 种颜色中的任一种。若使用单色监视器或黑白电视机，这 16 种颜色就变为黑白两色及三种不同的

灰度，色块之间全部相连。

低分辨率图形显示数据所占的存储区与40列文本显示数据所占的区域相同，每个色块占一个字符宽度和半个字符高的位置，因此每个字节表示两个色块，一个在上，一个在下。

显示数据的每半个字节(或四位)表示一个色块，低4位表示上面的色块，高4位表示下面的色块，如图2.2。4位二进制值表示16种不同的颜色，每种颜色4位数的关系如表2.2。

在每个字节中，低4位设置上面色块的颜色，高4位设置下面色块的颜色。如某显示数据为$DC，那么上面色块为绿色；下面色块为浅橙色。

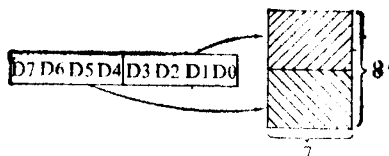


图2.2 低分辨图形显示数据与屏幕位置示意图

表 2.2 低分辨率图形彩色码

十进制	十六进制	颜色	十进制	十六进制	颜色
0	$0	黑	8	$8	棕
1	$1	深红	9	$9	橙
2	$2	深蓝	10	$A	灰 2
3	$3	紫	11	$B	粉红
4	$4	深绿	12	$C	浅绿
5	$5	灰 1	13	$D	黄
6	$6	中蓝	14	$E	海蓝
7	$7	浅蓝	15	$F	白

三、高分辨率图形方式

高分辨率图形显示 280×192 个彩色点的点阵，颜色有黑、白、紫、绿、橙、蓝共 6 种，颜色相同的相邻点形成一片显示区域。

高分辨率图形显示数据存储在两个 8192 个字节的存储区中，这两个存储区分别叫作显示页面 1 和显示页面 2。一般情况下，是使用高级语言来作图，在此只描述高分辨图形显示数据在缓冲区中的存储方式。

高分辨图形显示是以屏幕上某个点对应于某存储单元字节中的某一位来作图的。每个显示字节的低 7 位决定了屏幕上一行相邻 7 个点的显示，屏幕上一行 280 个点的显示由相邻 40 个存储单元来决定。如图 2.3 所示，在每个显示字节中，最低位决定了相邻 7 点中最左边点的显示，次低位决定了较右一点显示，以此类推，第 6 位决定了 7 点中最右边点的显示，两者的次序正好相反。显示数据的第 7 位不显示，用来选择两组颜色中的一组，如表 2.3。

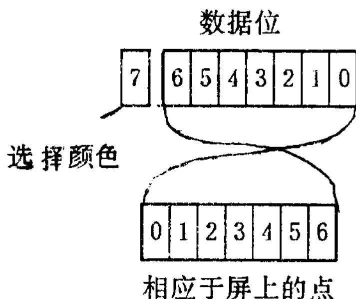


图 2.3 数据位与显示点的对应关系

表 2.3　高分辨率图形显示颜色选择

位 0～6	位 7 为 0	位 7 为 1
相邻列为 0	黑	黑
偶数列为 1	紫	蓝
奇数列为 1	绿	橙
相邻列为 1	白	白

四、中文方式

中文显示采用了高分辨率图形显示方式，能显示 16×16 点阵构成的 7445 种图形字符。其中包括 6763 个汉字及英文字母、日文假名、拉丁字母、俄文字母、希腊字母等。显示屏上共 11 条，由于第 11 行为状态行用来显示当前的输入方式和相应的提示信息，因此满屏幕实际显示 10 行，每行 17 个汉字，共计 170 个汉字。在中文方式下，处于“字母”方式（即显示西文）时，输入的英文字母和各种符号，在屏幕上占半个汉字的位置，即全屏幕 10 行每行 34 个字符。如处于“区位”或“拼音”方式时，按区位码输入各种字母和符号时，在屏幕上占一个汉字的位置、即全屏幕 10 行，每行 17 个字符。

五、混合方式

图形和文本同时在屏幕上显示称为混合方式。低分辨图形显示是色块组成的阵列，色块数为 40（宽）×48（高），在混合方式时，四行文本占了 8 行色块空间，剩下了 40×40 个色块空间。高分辨图形显示是由点阵组成的，点的个数为 280（宽）×192（高）个，在混合方式下，4 行文本占去了 32 行点，

剩下了 280×160 个点的区域。

六、显示缓冲区

中华学习机使用指定的缓冲区存储显示数据，这些存储区叫显示缓冲区或显示页面。显示缓冲区中每个字节控制屏幕上某一位置的显示内容。在文本方式中控制是一个字符，在低分辨方式中控制一个上下两部分的色块，在高分辨率图形中控制一行中相邻的 7 个点。

在 40 列文本及低分辨率图形方式中，有两个 1024 字节的显示页面，显示页面 1 地址为 $400— $7FF，显示页面 2 地址为 $800— $BFF。一般情况下仅使用页面 1，也可以通过软开关在页面 2 中存放文本或数据。这两个页面都可以用作 40 列文本、低分辨率图形及混合方式显示。

高分辨图形方式有两个 8192 字长的显示面，在 40 列文本及低分辨图形方式中，每个显示数据控制 7×8 点的区域，而在高发辨图形方式中，每个字节控制 1×7 个点，因而需要的显示缓冲区是 40 列文本的 8 倍，显示页面 1 地址为 $2000～ $3FFF，显示页面 2 地址为 $4000～ $5FFF。请参看表 2.4。

表 2.4　显示缓冲区地址表

显示方式	页　面	地　　址
40 列文本及	1	$400～ $7FF　1024～2047
低分辨图形	2	$800～ $BFF　2048～3071
高分辨图形	1	$2000～ $3FFF　8192～16383
	2	$4000～ $5FFF　16384～24575

七、显示方式的转换

要选择不同的显示方式，可通过软开关进行转换，也就是读或写专门的存储单元。大多数软开关有三个单元，一个用于接通软开关，一个用于断开软开关，一个用于读软开关状态。当读软开关时，可得到相应软开关地址的数据，此数据仅最高位 MD7(第 7 位)有效。在机器语言程序中，第 7 位为 1 表示此软开关 ON，第 7 位为 0 表示软开关 OFF。在 BASIC 程序中，得到的数值大于或等于 128 表示此软开关 ON，小于 128 数值表示软开关 OFF。

表 2.5 列出了控制不同显示方式选择的软开关。例：在使用汇编语文时要把混合方式转化为全屏幕图形显示方式，使用如下指令：

```
STA $C052
```

而在 BASIC 程序中使用如下命令：

```
POKE  49243,0
```

在使用各种高级语言情况下，显示方式的选择可以自动进行，而不需直接去读写表中单元. 所有显示方式都使用同一种基本地址分配方法：即字或图形数据按行存储，每行占连续的 40 个字节。但显示存储器地址与显示器上的位置并非按次序相应的，而实际上显示器上的位置是变化的(如图 2.5)。以 3 行为 1 组，每组中显示存储器相邻两行数据在屏幕上显示位置相隔 8 行。此组占存储器中每块(128 个字节)中开头的 120 个字节。这一方式叫折叠显示数据在存储器中的存储方式，利用这一方式在 1K 字节中存储 960 个显示数据。

表 2.5　显示软开关

名 称	功 能	单元 Hex	Dec		注释
ALTCHA	替代字符组接通	$C00F	49167	-16369	写
RSET	替代字符组断开	$C00E	49166	-16370	写
	读 ALTCHARSET 开关	$C01E	49182	-16354	读
TEXT	文本方式接通	$C051	49233	-16303	
	文本方式断开(图形)	$C050	49232	-16304	
	读 TEXT 开关	$C01A	49178	-16058	读
MIXED	混合方式接通	$C053	49253	-16301	1
	混合方式断开	$C052	49243	-16302	1
	读 MIXED 开关	$C01B	49179	-16057	读
PAGE2	页面 2 接通	$C055	49237	-16299	2
	页面 2 断开(页面 1)	$C054	49236	-16300	2
	读 PAGE2 开关	$C01C	49180	-16356	读
HIXES	高分辨方式接通	$C057	49239	-16297	1
	高分辨方式断开	$C056	49238	-16298	1
	读 HIRES 开关	$C01D	49198	-16355	读

注释 1:这种方式仅在图表方式 ON 有效。

注释 2:在 80 列卡的辅助文本页面被写使能时,此开关具有另一种功能。

在使用各种高级语言情况下,显示方式的选择可以自动进行,而不需直接去读写表中的单元。

八、显示页的结构

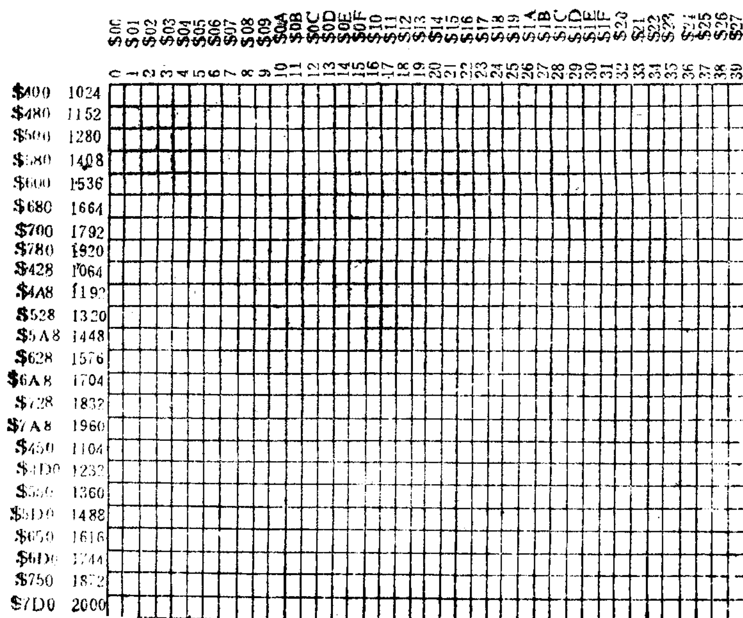


图 2.5　40 列文本显示页结构图

低分辨图表显示页结构图

2.2 倍高分辨率图形方式

如果 CEC—2000 型中华学习机插上专用内存扩展卡，且只需配上辅存 0 体的 64KRAM，就可使系统不仅内存被扩充到 128KRAM，同时具有倍高分辩率图形显示方式，分辩率为 560×192，颜色达到 16 种。可运行 APPLE Ⅱ e 的倍高分辩率图形精采游戏，方法是：开机后，立即按一下（ESC）键（必须在驱动器工作指示灯亮之前按下），则系统进入 APPLE Ⅱ e 兼容

工作方式。

倍高分辩率图形显示方式的使用较为复杂，用户可参考有关 APPLE Ⅱ e 的书及文章。下面只给出一个简单例子供参考。

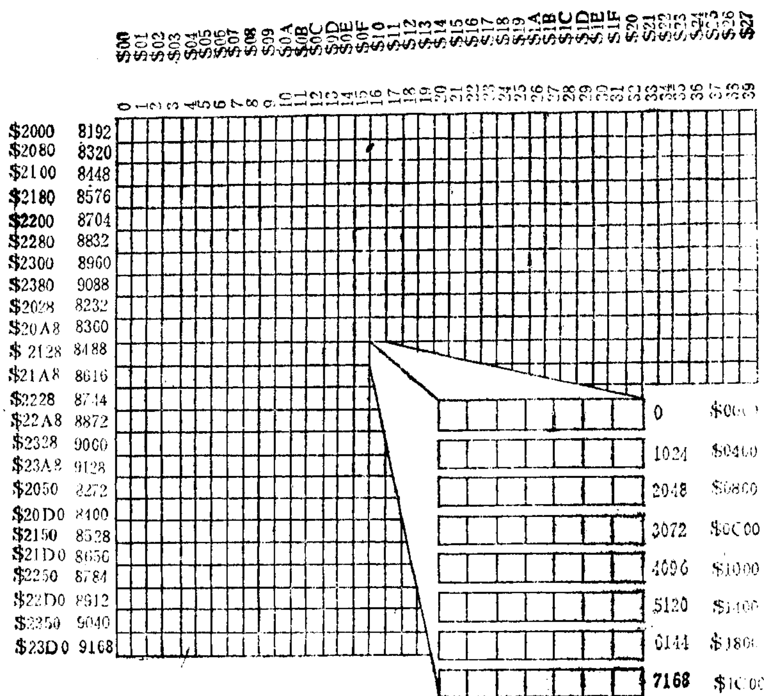


图 2.6 高分辨图形显示页结构图

在监控中切不述软开关后，可使 CEC—2000 进入倍高分辩率图形方式：

STA $ C050；置图形方式

```
STA $C052;非混合方式
STA $C054;第一页
STA $C057;高分辨率方式
LDA# $00
STA $C071;置 RAM 辅存第零体
STA $C00D;置 80 例方式
STA $C05E;置倍高分辨率。
```

进入倍高分辨图形方式后，图形数据缓冲区仍用 $2000～$3FFF(第一页)或 $4000～$5FFF(第二页)。但数据要在辅存和主存上交替存放。每一个地址上的 8 位数据确定了屏幕上的 7 个象素(点)。数据的最低位对应于屏幕左边的一个点，次高位对应右边的一个点。最高位不起作用。在一条水平线上共有 7×80＝560 个点。

倍高分辨率可在屏幕上产生 16 种颜色，它比普通高分辨率(192×280)的显示方式的颜色增加了许多种，而与低分辨率图形方式的颜色一样多。

倍高分辨率图形的颜色可以用下述方法来确定：在每一行上可以存放 80 个字节的数据，第 0 个字节放在辅存，第二个字节放在主存…。第七九个字节在主存。每一个字节数据中的 4 位(bit)确定了某种颜色。然而到底确定了那种颜色还必须看它们所处的位置。我们用该字节的位置除以 4，就会得到一个余数(只能是 0.1.2.3 这四个数之一)。用这个余数去查表一就能知道在这个字节里放什么样的数据，就能得到想要的颜色了。

当余数为偶数时，数据应放到辅存，余数为奇数时，数据应放到主存。由于每个字节仅有 7 个点可以显示，而一个彩

色点要由 4 位来确定，所以要显示 2 个以上的彩色点就至少

表一

颜色编号	颜色	余数			
		0	1	2	3
0	黑	$00	$00	$00	$00
1	深红	$88	$11	$22	$44
2	深蓝	$11	$22	$44	$88
3	紫	$99	$33	$66	$CC
4	深绿	$22	$44	$88	$11
5	灰 1	$AA	$55	$AA	$55
6	中蓝	$33	$66	$CC	$99
7	浅蓝	$BB	$77	$EE	$DD
8	棕	$44	$88	$11	$22
9	橙	$CC	$99	$33	$66
10	灰 2	$55	$AA	$55	$AA
11	粉	$DD	$BB	$77	$EE
12	浅绿	$66	$CC	$99	$33
13	黄	$EE	$DD	$BB	$77
14	海蓝	$77	$EE	$DD	$BB
15	白	$FF	$FF	$FF	$FF

与两个字节的数据有关。在一行内可显示的彩色点的数目为 560÷4=140 个。表二所列的程序是用 0～15 号颜色依次涂满整个屏幕。

表二

```
0300—  8D  50  C0      STA   $C050
0303—  8D  57  C0      STA   $C057
0306—  8D  52  C0      STA   $C052
0309—  8D  54  C0      STA   $C054
030C—  8D  0D  C0      STA   $C00D
030F—  8D  5E  C0      STA   $C05E
0312—  A2  00          LDX   #$00
0314—  A0  00          LDY   #$00
0316—  A9  20          LDA   #$20
0318—  84  00          STY   $00
031A—  85  01          STA   $01
031C—  8A              TXA
031D—  85  02          STA   $02
031F—  0A              ASL
0320—  0A              ASL
0321—  0A              ASL
0322—  0A              ASL
0323-  05  02          ORA   $02
0325—  4A              LSR
0326—  90  02          BCC   $032A
0328—  09  80          RA    #$80
032A—  A0  00          LDY   #$00
032C—  8D  05  C0      STA   $C005
032F—  91  00          STA   ($00),Y
0331—  C9  80          CMP   #$80
0333—  2A              ROL
0334—  8D  04  C0      STA   $C004
0337—  91  00          STA   ($00),Y
0339—  C9  80          CMP   #$80
033B—  2A              ROL
033C—  C8              INY
033D—  D0  ED          BNE   $032C
033F—  E6  01          INC   $01
0341—  A4  01          LDY   $01
0343—  C0  40          CPY   #$40
0345—  90  E3          BCC   $032A
0347—  8D  10  C0      STA   $C010
034A—  2C  00  C0      BIT   $C00
034D—  10  FB          BPL   $034A
034F—  E8              INX
0350—  E010            CPX   #$10
0352—  90  C0          BCC   $0314
0354—  B0  BC          BCS   $0312
```

第三章　输入系统

3.1 键盘工作原理

键盘的读过程

从执行机器语言的角度上来看，要得到键盘上某个键的字符编码，是通过读某个存储器专用单元的内容而实现的。这些专用单元的地址及功能如下表。

表 3.1　键盘存储单元

十六进制	十进制	单元内容说明
$C000	45152,－16384	键盘数据及选通单元
$C010	49168,－16368	键按下标志及清除选通单元

用程序读键盘存储单元，就可以得到最后按下键的键码。$C000 单元的低七位是当前按下键的字符码，最高位是选通位。在 BASIC 程序中用 PEEK 命令来取得这一状态信息，若取得值大于等于 128，表示键已按下，若小于 128，表示无键按下。

当某键按下时，键盘数据及选通单元（$C000）的最高位为 1，此位状态一直保持到读或写 $C010 单元（键按下标志及清除选通单元）为此。$C010 单元包括键按下和清除选通开关两部分。由于此单元清除选通开关功能是由软件来实现

的(即对此单元进行读或写操作),因而也叫软开关。无论读或写此单元,这一开关功能都会使键盘数据及选通单元($C000)的选通位复位。因而当程序既要判断是否有键按下,又要看脉冲选通位是否为高,必须首先读$C000,然后再读$C010,否则脉冲选通位将被复位。当选通单元清除后一直保持为低,直到又有键按下。在选通位复位而没有另一键按下时键盘数据没有变化,因而仍可读此键盘数据。

3.2 键对应的 ASCII 码表

表中有些键如 CONTROL,SHIFT,CAPSLOCK 单独使用时并不产生 ASCII 码,而与其它 ASCII 键同时按下时产生另一些 ASCII 码(有些保持原来的 ASCII 码)。RESET 键直接连到主机上,它和 CONTROL 键同时按下产生复位。

表 3.2　键对应的 ASCII 码表

键符	Normal	Shift	Control	both
) 0	30	29	30	29
! 1	31	21	31	21
@ 2	32	40	32	40
# 3	33	23	33	23
$ 4	34	24	34	24
% 5	35	25	35	25
^ 6	36	5E	1E	1E
& 7	37	26	37	26
# 8	38	2A	38	2A
(9	39	28	39	28
—	2D	5F	2D	1F
+ =	3D	2B	3D	2B
| 、	5C	7C	1C	1C
{ [	5B	7B	1B	1B
}]	5D	7D	1D	1D

续表

键符	Normal	Shift	Control	both
: ;	3B	3A	3B	3A
" '	27	22	27	22
~ •	60	7E	60	7E
> ,	2C	3C	2C	3C
< 、	2E	3E	2E	3E
? /	2F	3F	2F	3F
Test	16	16	16	16
A	61	41	01	01
B	62	42	02	02
C	63	43	03	03
D	64	44	04	04
E	65	45	05	06
F	66	46	06	06
G	67	47	07	07
H	68	48	08	08
I	69	49	09	09
J	6A	4A	0A	0A
K	6B	4B	0B	0B
L	6C	4C	0C	0C
M	6D	4D	0D	0D
N	6E	4E	0E	0E
O	6F	4F	0F	0F

续表

键符	Normal	Shift	Control	both
P	70	50	10	10
Q	71	51	11	11
R	72	52	12	12
S	73	53	13	13
T	74	54	14	14
U	75	55	15	15
V	76	56	16	16
W	77	57	17	17
X	78	58	18	18
Y	79	59	19	19
Z	7A	5A	1A	1A
中文(F_6)	1F	1F	1F	1F
西文	7F	7F	7F	7F
F_1	01	01	01	01
F_2	16	16	16	16
F_3	17	17	17	17
F_4	14	14	14	14
F_5	06	06	06	06
ESC	1B	1B	1B	1B
TAB	09	09	09	09
Break	03	03	03	03
Return	0D	0D	0D	0D
空格	20	20	20	20
←	08	08	08	08
↑	0B	0B	0B	0B
↓	0A	0A	0A	0A
→	15	15	15	15

4.1 盒式录音机接口电路

在 CEC—2000 上，配有盒式录音机接口。利用该接口可将普通的盒式录音机作为学习机的外部存储器，把 RAM 中的信息记录在磁带上，保存起来。还可以把存在磁带上的程序或数据重新装入 RAM 之中。

CEC—2000 的盒式录音机接口，采用硬件进行信号电平匹配，波形的整形。由软件完成数据的调制和解调，进行数据记录格式的管理。接口中，输出给录音机的信号由 IOU 的软开关 CASSO 产生，并由输出端 CASSO 输出。IOU 输出的信号经电阻网络分压后，作为给录音机话筒输入(MIC)的录音信号，将调制信号记录在磁带上。读出信息时，录音机的耳机插孔(EAR)输出信号，作为接口的输入信号，输入信号经运算放大器 U15 判别整形后，送到数据选择器 U4 的输入端 10，读 $C060 单元时，就可以对当前录音机输出信号进行采样。

录音机接口信号要求：

输入信号：录音机的耳机插孔(AER)输出信号峰—峰电压值应不小于 1V，输入阻抗约为 12K。

输出信号：给录音机话筒输入(MIC)的输出信号约为 25mV，输出阻抗约为 100Ω。

4.2 游戏杆接口电路

一、游戏开关

在游戏接口电路中，有三个开关量的输入称为游戏开关SW0、SW1、SW2。它们都接到数据选择器U4的输入端。在这里，地址$C06×的译码信号作为U4的使能信号，地址A0—A2作为选择控制。根据地址A0—A2的不同组合，U4可以选择某一个输入作为输出。SW0、SW1、SW2分别接在U4的I1、I2和I3，读取SW0—SW2状态的地址如表4.1所示。

游戏开关的典型接法如图4.1。

这是以输入SW0为便，对SW1和SW2可以采用同样的方法。

当开关K未被按下时，电阻将SW0拉为低电平。若此刻访问$C061，则可以从数据线上读得D7=0。当开关K被按下时，SW0被接到+5V，若访问$C061，则可以从数据线上读得D7=1。这样，就可以测出SW0的状态。

表4.1 读游戏开关地址表

地　　址	游戏开关
$C061	SW0
$C069	
$C062	SW1
$C06A	
$C063	SW2
$C06B	

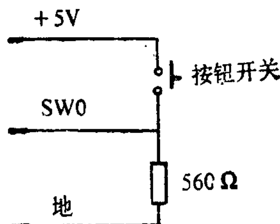


图 4.1　游戏开关接法(以 SWO 为例)

二、游戏摇杆

在游戏接口中,有 4 路模拟量输入,它们是游戏摇杆输入。

在游戏接口中,每一路摇杆输入都是通过一个 150kΩ 的电位器接到+5V。时基电路输出的正脉冲宽度由 6502 来测量,地址 $C07×的译码信号作为时基电路 558 的触发信号。对这 4 路输出状态的读取方法和 SW 类似。读取这 4 路输出状态的地址如表 4.2 所示。

表 4.2　读摇杆信号地址表

地　　址	摇杆信号
$C064 $C06C	PDL0
$C065 $C06D	PDL1
$C066 $C06E	PDL2
$C067 $C06F	PDL3

对于 558 的每一部分时基电路，其定时电容都为 0.022μf，定时电阻由主板上 100Ω 的电阻和摇杆中的电位器串联而成。摇杆中的电位器阻值一般为 150kΩ，这样，可以得到时基电路输出的最宽的正脉冲宽度为：

$$(150k+100)\Omega \times 0.022\mu f \doteq 3.3ms$$

最窄的正脉冲宽度为：

$$100\Omega \times 0.022\mu f = 2.2\mu s$$

4.3 扬声器接口电路

在学习机的机盒内，有一只 0.25W8Ω 的扬声器。在 IOU 内的一个软开关控制下，扬声器可以发出各种响声。IOU 对扬声器的控制输出端是 SPKR，它输出的信号加到 Q3 的基极。Q3 是达林顿三极管，它放大的信号经电阻分压后，驱动扬声器。

控制扬声器的软开关 SPKR，对应的地址为 $C03×，该软开关有两个状态："0"和"1"。对于输出端 SPKR，分别输出高电平和低电平。每当访问 $C030 时，软开关的状态翻转一次，即状态由"1"变为"0"或者由"0"变"1"。要让扬声器发出可以听到的响声，就需要有一个持续一定时间长度，其频率在音频范围内的信号去驱动扬声器。信号持续的时间长短决定发出响声的长短，声音的音调取决于信号的频率。

若要得到某种音调的声音，首先要计算好该音调的频率和周期。编制程序，进行半周期时间长度的延迟。每当半个周期时间延迟结束时，对地址 $C030 访问一次，使软开关状态

翻转一次，然后再进行半周期的延迟。将这种操作循环进行下去，循环上面操作的时间就是扬声器发出所要音调响声时间长度。

4.4 软开关

1. 存储器管理部件(MMU)

MMU 负责管理学习机中的存储器系统。电路内设有软开关。根据软开关状态的设置，可进行主存储器和辅存储器的切换；进行存储体的选择和显示缓存区的页面选择。另外，MMU 还将 CPU 的 16 位地址转换成行列地址，并将其分时输出。MMU 可以根据其内部软开关的状态以及 CPU 输出的地址，确定 CPU 访问哪一种存储器空间，并输出相应的选择控制信号

2. 软开关

在 MMU 中设有软开关，供用户使用。利用这些软开亲，可使 MMU 处于不同的工作方式中，并按使用者的意图，对存储体空间进行多种分配一组合，使存储空间得到灵活使用。在 MMU 中，软开关大致可分为四大类：(1)主存/辅存切换控制；(2)内部 ROM 空间和 I/O 空间的切换；(3)空间 \$D000—\$FFFF 的选择；(4)显示缓冲区页面的选择。

1. 主存/辅存切换控制

这类开关有4个，利用对指定地址进行读操作，完成对软开关的设置。软开关与相应地址的对应关系见表4.3。

表4.3　主/辅存切换软开关地址表

软开关	置　位	复　位	读状态
80STORE	$C001	$C000	$C018
RAMRD	$C003	$C002	$C013
RAMWRT	$C005	$C004	$C014
ALTZP	$C009	$C008	$C016

80STORE：控制主存/辅存切换空间区段的范围。

为ON时，可根据软开关HIRES、PAGE2的不同设置选择文本显示缓冲区第一页$0400—$07FE和高分辩图形缓冲区第一页$2000—$3FFF在主存或在辅存之中。

为OFF时，则存储空间$0200—$BFFF在主存/辅存空间的切换。完全受RAMRD和RAMWRT控制。

RAMRD：选择读操作，在主存或在辅存进行。

为ON时，选择对空间 $ 0200— $ BFFF的读操作，在辅存进行。

为OFF时，选择对空间 $ 0200— $ BFFF 的读操作，在主存进行。

RAMWRT：选择写操作，在主存或在辅存进 行。

为ON时，对空间 $ 0200— $ BFFF 的写操作在辅 存 进行。

为OFF时，对空间 $ 0200— $ BFFF 的写操作在主存进行。

ALT2P：选择体RAM 在主存或在辅存。

为 ON 时，将 $ 0000 — $ 01FF 和 体 RAM $ D000— $ FFFF映射到辅存。

为OFF时，这两块空间映射在主存上。

2．内部ROM空间和I/O空间切换控制

这类软开关有两个，它们利用写操作完成其设置。见表4.4

表 4.4　内部ROM和I/O空间切换软开关地址表

软开关	置位	复位	读状态
SLOTCXROM	$ C007	$ C006	$ C015
SLOTC3ROM	SC00B	$ C00A	$ C017

SLOTCXROM：为ON时，空间 $ C100— $ CFFF 属 于内部ROM。

为OFF时，对空间＄C100—＄CFFF的访问在I/O ROM里进行。

SLOTC3ROM：为ON时，空间C3××映射到I/O ROM，即这256个字节属于槽口ROM。

为OFF时，空间C3××映射到内部ROM。

软开关SLOTCXROM和SLOTC3ROM都可控制空间＄C3××在I/O空间与内部ROM之间的切换。只要这两个软开关中有一个的状态是置空间＄C3××在内部ROM，那么＄C3××这一页空间就在内部ROM内。要将空间＄C3××切到I/O空间，则要求这两个软开关的状态都是置＄C3××这一页在I/O空间。

3．为空间＄D000—＄FFFF的选择控制

空间＄D000—＄FFFF可以映射到ROM区或者RAM区；也可以将读操作映射到ROM区，同一地址的写操作映射在RAM区；还可以对RAM进行写保护。另外，空间＄D000—＄DFFF还可以映射到两个4K的RAM存储体上。对这些使用方式的选择，是通过访问＄C080—＄C08F这16个单元来完成的。

为了构成对地址访问的各种组合，设有三个状态：ROM读使能，RAM写使能，RAM第一体。ROM读使能为“ON”时，可以读ROM；为“OFF”时，可以读RAM。RAM写使能为“ON”时，可以写RAM；为“OFF”时，禁止写RAM即RAM写保护。RAM第一体为“ON”时，选择4KRAM第一体；为“OFF”时，选择RAM第二体。

进行选择控制时，要求对一个地址进行访问时，就能完成一种使用方式的选择。即设置上面讲的三种状态为当前值。这里所给地址为＄C080—＄C08F。故有4位地址A3—A0作为状态设置位。

ROM读使能信号为：A0⊕A1。

RAM写使能信号为：A0。

RAM第一体信号为：A3。

4. 显示缓存区页面选择软开关

软开关HIRES、PAGE2和80STORE的各种状态组合，可以选择显示缓存区空间映射在主存还是在辅存，以及当前屏幕的显示缓存区是在显示第1页面还是第2页面。

软开关HIRES和PAGE2是通过访问如表4.5所示的地址完成其状态的复位和置位，并可读出软开关的当前状态。

当80STORE状态为"ON"时，显示缓存区的第1页可以在PAGE2的选择下，映射在主存或辅存。PAGE2所能选择的页面大小由HIRES确定。

表4.5 软开关HIRES、PAGE2地址表

软开关	置位地址	复位地址	读状态
HIRES	＄C057	＄C056	＄C01D
PAGE2	＄C055	＄C054	＄C01C

注：这两个软开关的状态实际上是从IOU读出。

若HIRES为"ON"时，PAGE2可以选择文本缓存区＄0400—＄07FF和高分辨图形缓存区＄2000—＄3FFF在主存或在辅存。若HIRES为"OFF"时，PAGE2仅能选择文本

缓存区＄0400—＄07FF在主存或在辅存。

当PAGE2为“ON”时，PAGE2所能选择的显示缓存区映射到辅存。当PAGE2为“OFF”时，相应的显示缓存区映射在主存中。

表4.6，给出了80STORE为“ON”时,HIRES和PAGE2的状态与显示缓存区地址映射的关系。

表4.6　显示缓存区主/辅存切换

80STORE	HIRES	PAGE2	文本显示缓存区 ＄0400-＄07FF	高分辨图形显示缓存区 ＄2000-＄3FFF
ON	OFF	OFF	主　存	主　存
ON	OFF	ON	辅　存	主　存
ON	ON	OFF	主　存	主　存
ON	ON	ON	辅　存	辅　存

当80STORE为“OFF”时，HIRES和PAGE2决定屏幕显示方式以及存放当前显示数据的显示缓存区的页面。关于这部分的详细介绍见“IOU的软开关”。

3.　输入输出部件（IOU）

输入输出部件控制屏幕显示，产生读显示缓存区的地址，并将读出的显示数据和视频显示控制信号一起送视频信号合成器，也产生给录音机和扬声器的输出信号。另外，还可以产生4个指示器输出信号。

为实现多种显式方式，并产生一系列的输出信号，IOU内设置了许多软开关。下面着重介绍IOU的关的功能。

4. 软开关

IOU内的软开关，根据其用途的不同，大致可以分为三类：显示方式控制；录音机、扬声器接口输出控制；指示器输出控制。

1. 显示方式控制

这类软开关有7个。通过对指定地址的访问，实现软开关的设置。有些软开关的状态必须由写操作设置。绝大部分软开关的状态都能读出。关于软开关与相应地址的关系见表4.7

表 4.7 控制显示方式软开关地址表

软开关	置位	复位	读状态	注
80STORE	$C001	$C000	——	1
80COI	$C00D	$C00C	$C01F	1
ALTCHARSET	$C00F	$C00E	$C01E	1
TEXT	$C051	$C050	$C01A	
MIXED	$C053	$C052	$C01B	
PAGE2	$C055	$C054	$C01C	
HIRES	$C057	$C056	$C01D	

注1：这些软开关状态的设置需对相应单元进行写操作。

关于软开关80STORE的功能已在MMU部分介绍了，这里就软开关控制显示方式的功能进行介绍时，将80STORE处于“OFF”状态。

80COL：选择80列字符显示工作方式。

为ON时，允许80列字符显示方式。

为OFF时，禁止80列字符显示方式。

ALTCHARSET：选择替代字符集。

为ON时，在文本方式中，所显示的字符属于替代字符集。

为OFF时，在文本方式中，所显示的字符属于基本字符集。

TEXT：选择文本显示方式。

为ON时，显示方式为文本方式，显示字符。

为OFF时，显示方式为图形方式，再根据HIRES的状态确定是低分辨图形显示还是高分辨图形显示。

MIXED：选择字符和图形混合显示方式。

为ON时，且TEXT为OFF，即处于图形显示方式，允许当前显示方式为字符和图形混合显示。

为OFF时，禁止字符和图形混合显示。

PAGE2：选择显示缓存区第2页面。

为ON时，当前屏幕显示的内容为显示缓存区第2页面的显示数据。

为OFF时，当前屏幕显示内容为显示缓存区第1页面的显示数据。

2. 扬声器、录音机输出控制

这部分软开关有两个，它们和前面讲的软开关有些不一样，即不能明确地设置当前的软开关的状态，只是访问指定地址一次，软开关状态翻转一次。软开关的当前状态也无法读出。软开关和对应地址关系见表 4.8

表 4.8 控制输出软开关地址表

软开关	访问地址
CASSO	$C02X
SPKR	$C03X

软开关 CASSO 控制产生给录音机的输出，通过控制访问地址 $C02X的时间间隔，得到所需宽度的正负方波输出。

软开关 SPKR 控制产生给扬声器的输出，也是通过控制访问地址 $C03X 的时间间隔，在其输出端得到规定频率的方波输出。

3．指示器输出控制

这部分有 4 个软开关，它们是通过对指定地址进行访问，来设置软开关的状态。它们当前状态，不能读取出来。软开关的相应地址见表 4.9 。

表 4.9 设指示器输出软开关地址表

软开关	置位	复位
AN0	$C059	$C058
AN1	$C05B	$C05A
AN2	$C05D	$C05C
AN3	$C05F	$C05E

这 4 个软开关 AN0、AN1、AN2、AN3 的状态分别在 IOU 的引脚 AN0、AN1、AN2、AN3 输出。

4.5 打印机部件

一 打印机接口

打印机接口卡是在主机送出的$\overline{\text{DEVIC}}$信号、$\overline{\text{I/OSEL}}$信号及$\overline{\text{I/OSTROBE}}$信号控制下，使主机上的EPROM工作，直接控制打印机。

当$\overline{\text{DEVSE1}}$低信号作用时，地址单元为：\$C090～\$C09F，由6502选址16个I/O地址中的一个。当$\overline{\text{I/OSEL}}$低信号作用时，同时可用地址的最低8位选择256个存储单元中的一个．地址单元为\$C100～\$C1FF。I/OSTROBE信号低作用时，选择扩充ROM空间地址为：\$C800—\$CFFF。

二　打印机的控制命令字符和软开关

打印机通过打印机接口与主机系统相连接后，成为受控于主机系统的一个外部设备。它具有接收并处理由主机送来的控制命令和打印数据的能力。通过打印机对这些命令的执行和处理，可以控制和改变打印数据的格式、走纸量和字符形状等等。还可以将输出的数据作为图象信息进行打印，以达到打印汉字的目的。

表4.10列出了打印机的控制命令字符表，这些字符可以用 BASIC 语言中的 PRINT CHR $（m）的语句送给打印机，以达到控制、指挥打印机动作的目的。其中（m）中的 m 是该命令字符的十进制数据代码。

表3.45列出了打印机的控制软开关，是指在主机系统中的打印机接口上所设置的单一地址单元，它是在进行打印时，由在该接口上的驱动控制软件进行识别处理的另一种控制打印机动作的方式。利用这些控制软开关进行打印机的动作控制，对程序员来说无疑是减轻了许多负担。

表4.11中，① SLOT 为打印机接口所占用的槽号，②控制硬拷贝的输出寄存器为 1912＋SLOT，该寄存器中的8位数据信息含义如下：

U	D	I	L			P	
行方式	放大	反相	异或	或	与	HI-RES 第二页	HI-RES 第一页
128	64	32	16	8	4	2	1

表4.10 打印机的控制命令字符表

	控制字符	十六进制代码	十进制代码	控制功能
打印动作	CTRL-M	0D	13	送回车字符，完成打印缓冲区数据打印
	CTRL-J	0A	10	完成打印，并走纸一行
	CTRL-K	0B	11	功能同CTRL-M
	CTRL-H	08	8	退格符，从打印缓冲区删去一个字符
	CTRL-G	07	7	响铃字符，使打印机鸣3秒
	CTRL-L	0C	12	走纸到下一页的页首
打印方式控制	CTRL-N	0E	14	置放大字符打印方式
	CTRL-O	0F	15	置压缩字符打印方式
	CTRL-T	14	20	清除放大字符打印方式
	CTRL-R	12	18	清除压缩字符打印方式
	CTRL-Q	11	17	启动高分辨率图象的硬拷贝
	CTRL-W	17	23	夹在二个CTRL-W间的字母按小写字母打印
走纸控制设置	NUL	00	00	设置命令的终结标志
	ESC 0	1B 30	27 48	设置行间隔为1/8英寸
	ESC 2	1B 32	27 50	设置行间隔为1/6英寸
	ESC A m	1B 41 m	27 65 m	设置行间隔为m/72英寸
	ESC 3 n	1B 33 n	27 51 n	设置行间隔为n/216英寸
	ESC C m	1B 43 m	27 67 m	设置每页的间隔行数，m在1～127之间

续表

	控制字符	十六进制代码	十制进代码	控　制　功　能
走纸控制设置	ESC C O m	1B 43 30 m	27 67 48 m	设置每页间的间隔为n时，n在1～22之间
	ESC D n_1 n_2 … nk 0	1B 44 n_1 n_2 … nR 33	27 68 n_1 n_2 … nk 48	设置k个表格定位，其中nk必须在1～233间，正常打印≤79，压缩打印≤131
打印控制位图象	ESC K n_1 n_2	1B 4B n_1 n_2	27 75 n_1 n_2	由文本打印变成标准密度的位图象打印
	ESC L n_1 n_2	1B 4C n_1 n_2	27 76 n_1 n_2	由文本打印变成倍密度位图象打印
	HT	09	09	移至打印头到表格指定位置，须与ESC D n_1 n_2 ……nk命令连用

当该寄存器中某一位为1时，使执行相应的操作。

其寄存器的各位含义如下：

（1）P参数为高分辨图形页选择。

P＝1　打印高分辨图形第一页

P＝2　打印高分辨图形第二页

P＝3　将第一、第二高分辨图形并列进行打印

（2）L参数决定二页高分辨图形逻辑操作。

L＝0　无逻辑操作

L＝4　进行与操作

表 4.11 打印机的控制软开关表

	软 开 关	控 制 功 能
1	POKE1656+SLOT,n	设置打印宽度(n个字符/行)
2	POKE1528+SLOT,255	禁止执行换行(LF)
	POKE1528+SLOT,0	释放LF
3	POKE1912+SLOT,M	设置高分辨率图形打印方式
	M=U+D+I+L+P	(U,D,I,L,P参数说明见后面的介绍)
4	POKE1272+SLOT,255	初始化接口
5	POKE1144+SLOT,76	设置倍密度位图象打印(960点/行)
	POKE1144+SLOT,75	设置标准密度位图象打印(480点/行)
6	POKE1400+SLOT YY	将控制字符变为小写字母

L=8 进行或操作

L=16 进行异或操作

(3) I 参数。

I=0 表示正向打印

I=32 表示反向打印

(4) D 参数。

D=0 正常打印

D=64 放大打印方式

(5) U参数。

U=0，图象方式，执行PRINT CHR$ (17)时，对整幅图象进行打印。

U=128，行方式，执行 PRINT CHR$ (17) 时，仅打

印该图象的一行。

由这些参数的总和便可获得所需要的硬拷贝方式寄存器的数值。

附录A 带有操作码，执行周期和存储要求的指令表

常用标志含义如下：

A	累加器
X，Y	变址寄存器
M	存储器
P	处理器状态寄存器
S	堆栈指针
✓	改变
—	不改
$+$	加
$\wedge$	逻辑与
$-$	减
$\forall$	异或
$\uparrow$	弹出堆栈
$\downarrow$	压入堆栈
$\rightarrow$	传送到
$\leftarrow$	传送到
$\vee$	逻辑或
PC	程序计数器
PCH	

PCL　　程序计数器低位

OPER　　操作数

#　　立即寻址方式

注解：在每张表的顶端插入一参考数字（Ref：××）用以直接表示用户在R6500微计算机系列程序设计手册中的有关章节，在那部分中指令被定义和被论述过。

ADC　带进位加　　ADC

N Z C I D V

操作：A+M+C→A，C　　✓✓ - - - -

寻址方式	汇编语言格式	操作码	字节数	周期数
立即型	ADC # oper	66	2	2
零页	ADC oper	65	2	3
零页X变址型	ADC oper, X	76	2	4
绝对型	ADC oper	6D	3	4
绝对X变址型	ADC oper, X	7D	3	4*
绝对Y变址型	ADC oper, Y	79	3	4*
变址间接型	ADC (oper, X)	61	2	6
间接变址型	ADC (oper, Y)	71	2	5*

* 如果有跨页则加一个周期

AND 逻辑与

累加器逻辑与

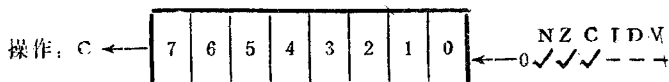
操作：A∧M→A

N Z C I D V
✓ ✓ — — — —

寻址方式	汇编语言格式	操作码	字节数	周期数
立即型	AND * oper	62	2	2
零页	AND oper	25	2	3
零页X变址型	AND oper，X	35	2	4
绝对型	AND oper	2E	3	4
绝对X变址型	AND oper，X	3D	3	4*
绝对Y变址型	AND oper，Y	39	3	4*
变址间接型	AND (oper，X	21	2	6
间接变址型	AND (oper). Y	31	2	5*

* 如果有跨页则加1个周期

ASL 左移



寻址方式	汇编语言格式	操作码	字节数	周期数
累加器型	ASL A	0A	1	2
零页	ASL oper	06	2	5
零页X变址型	ASL oper，X	16	2	6
绝对	ASL oper	0E	3	6
绝对X变址型	ASL oper，X	1E	3	7

BCC进位为"0"分枝 BCC

操作：C=0 分枝

N Z C I D V
— — — — — —

寻址方式	汇编语言格式	操作码	字节数	周期数
相对型	BCC oper	90	2	2*

* 如果分枝发生在同页上则加1。
* 如果分枝发生在不同页上则加2。

BCS 进位为"1"分枝 BCS

操作：C=1 分枝

N Z C I D V
— — — — — —

寻址方式	汇编语言格式	操作码	字节数	周期数
相对型	BCS oper	B0	2	2*

* 如果分枝发生在同一页上则加1。
* 如果分枝发生在下一页上则加2。

BEQ 结果为"0"分枝 BEQ

操作：Z=1 分枝

N Z C I D V
— — — — — —

寻址方式	汇编语言格式	操作码	字节数	周期数
相对型	BEQ oper	F0	2	2*

* 如果分枝发生在同一页上则加1。
* 如果分枝发生在下一页上则加2。

BIT 位测试 BIT

NZCIDV

操作：A∧M，$M_7 \to N$，$M_6 \to V$　　　　M_7✓ — — — M_6

位 6 和位 7 被传送到状态寄存器

如果 A∧M 的结果是零则 Z=1，否则 Z=0。

寻址方式	汇编语言格式	操作码	字节数	周期数
零　页	BIT　oper	24	2	3
绝对型	BIT　oper	2C	3	4

BMI 结果为负分枝 BMI

NZCLDV

操作：N=1 分枝

寻址方式	汇编语言格式	操作码	字节数	周期数
相对型	BMI　oper	30	2	2*

* 分枝发生在同一页上则加 1。
* 分枝发生在不同页上则加 2。

BNE 结果不为"0"分枝 BNE

NZCIDV

操作：Z=0 分枝　　　　— — — — — —

寻址方式	汇编语言格式	操作码	字节数	周期数
相对型	BNE　oper	D0	2	2*

* 如果分枝发生在同一页上则加 1。
* 如果分枝发生在不同页上则加 2。

N Z C I D V
— — — — — —

操作：N=0 分枝

寻址方式	汇编语言格式	操作码	字节数	周期数
相 对 型	BPL oper	10	1	2

• 如果分枝发生在同页上则加 1。
• 如果分枝发生在不同页上则加 2。

BRK 强制暂停 BRK

N Z C I D V
— — — 1 — —

操作：强制中断 PC+2↓P↓

寻址方式	汇编语言格式	操作码	字节数	周期数
隐 含 型	BRK	00	1	7

• BRK命令不能用置 I 位屏蔽

BVC 溢出为"0"分枝 BVC

N Z C I D V
— — — — — —

操作：V=0 分枝

寻址方式	汇编语言格式	操作码	字节数	周期数
相 对 型	BVC oper	50	2	2*

• 如果分枝发生在相同的页则加 1。
• 如果分枝发生在不同的页则加 2。

BVS 溢出为"1"分枝 BVS

操作：V=1 分枝

N	Z	I	C	D	V
—	—	—	—	—	—

寻址方式	汇编语言格式	操作码	字节数	周期数
相对型	BVS oper	70	2	2*

* 如果分枝发生在同一页上则加 1。
* 如果分枝发生在不同页上则加 2。

CLC 清进位标志 CLC

操作：0→C

N	Z	C	I	D	V
—	—	0	—	—	—

寻址方式	汇编语言格式	操作码	字节数	周期数
隐含型	CLC	18	1	2

CLD 清十进制方式 CLD

操作：0→D

N	Z	C	I	D	V
—	—	—	—	0	—

寻址方式	汇编语言格式	操作码	字节数	周期数
隐含型	CLD	D8	1	2

CLI 清中断禁止位 CLI

操作：0→I

N Z C I D V
— — — 0 — —

寻址方式	汇编语言格式	操作码	字节数	周期数
隐 含 型	CLI	58	1	2

CLV 清溢出标志 CLV

操作：0→V

N Z C I D V
— — — — — 0

寻址方式	汇编语言格式	操作码	字节数	周期数
隐 含 型	CLV	B8	1	2

CMP 累加器与存储器比较 CMP

操作：A—M

N Z C I D V
✓ ✓ ✓ — — —

寻址方式	汇编语言格式	操作码	字节码	周期码
立 即 型	CMP *oper	C9	2	2
零 页	CMP oper	C5	2	3
零页X变址型	CMP oper，X	D5	2	4
绝 对 型	CMP oper	CD	3	4
绝对X变址型	CMP oper，X	DD	3	4*
绝对Y变址型	CMP oper，Y	D9	3	4*
变址间接型	CMP (oper，X)	C1	2	6
间接变址型	CMP (oper)，Y	D1	2	5*

* 如果跨页则加 •

CPX 变址寄存器X与存储器比较

操作：X—M

N Z C I D V
✓ ✓ ✓ — — —

寻址方式	汇编语言格式	操作码	字节数	周期数
立即型	CPX #oper	EO	2	2
零页	CPX oper	E4	2	3
绝对值	CPX oper	EC	3	4

CPY 变址寄存器 Y 与存储器比较

操作：Y—M

N Z C I D V
✓ ✓ ✓ — — —

寻址方式	汇编语言格式	操作码	字节数	周期数
立即型	CPY #oper	C0	2	2
零页	CPY oper	C4	2	3
绝对型	CPY oper	CC	3	4

DEC　存储器减 1

操作：M—1→M

N Z C I D V
✓✓— — — —

寻址方式	汇编语言格式	操作码	字节数	周期数
零　　页	DEC　oper	C6	2	5
零页X变址型	DEC　oper，X	D6	2	6
绝　　对	DEC　oper	CE	3	6
绝对X变址型	DEC　oper，X	DE	3	7

DEX　变址寄存器减 1

DEX

操作：X—1→X

N Z C I D V
✓✓— — — —

寻址方式	汇编语言格式	操作码	字节数	周期数
隐 含 型	DEX	CA	1	2

DEY　变址Y寄存器减 1

DEY

操作：Y—1→Y

N Z C I D V
✓✓— — — —

寻址方式	汇编语言格式	操作码	字节数	周期数
隐 含 型	DEY	88	1	1

EOR 逻辑异或

操作：A∀M→A

N Z C I D V
✓ ✓ — — — —

寻址方式	汇编语言格式	操作码	字节数	周期数
立即型	EOR #oper	49	2	2
零页	EOR oper	45	2	3
零页X变址型	EOR oper X	55	2	4
绝对	EOR oper	4D	3	4
绝对X变址型	EOR oper, X	5D	3	4*
绝对Y变址型	EOR oper, Y	59	3	4*
变址间接型	EOR (oper, X)	41	2	6
间接变址型	EOR (oper), Y	51	2	5*

* 如果跨页则加1。

INC 存储器加1

操作：M+1→M

N Z C I D V
✓ ✓ — — — —

寻址方式	汇编语言格式	操作码	节字数	周期数
零页	INC oper	E6	2	5
零页X变址型	INC oper, X	F6	2	6
绝对	INC oper, X	EE	3	6
绝对X变址型	INC oper, X	FE	3	7

INX 变址寄存器X加1 INX

操作：X+1→X

N Z C I D V
✓ ✓ — — — —

寻址方式	汇编语言格式	操作码	节字数	周期数
隐 含 型	INX	E8	1	2

INY 变址寄存器Y加1 INY

操作：Y+1→Y

N Z C I D V
✓ ✓ — — — —

寻址方式	汇编语言格式	操作码	字节数	周期数
隐 含 型	INY	C8	1	2

JMP 转移 JMP

操作： (PC+1)→PCL
(PC+2)→PCH

N Z C I D V
— — — — — —

寻址方式	汇编语言格式	操作码	字节数	周期数
绝 对 型	JMP oper	4C	3	3
间 接 型	JMP (oper)	6C	3	5

JSR 转子

操作：PC+2↓，(PC+1)→PCL
(PC+2)→PCH

N Z C I D V
— — — — — —

寻址方式	汇编语言格式	操作码	字节数	周期数
绝对型	JSR oper	20	3	6

LDA 送累加器

操作：M→A

N Z C I D V
✓ ✓ — — — —

寻址方式	汇编语言格式	操作码	字节数	周期数
立即型	LDA #oper	A9	2	2
零页	LDA oper	A5	2	2
零页X变址型	LDA oper X	B5	2	4
绝对型	LDA oper	AD	3	4
绝对X变型址	LDA oper, X	BD	3	4*
绝对Y变址型	LDA oper, Y	B9	3	4*
变址间接型	LDA (oper, X)	A1	2	6
间接变址型	LDA (oper), Y	B1	2	5*

* 如果跨页则加1。

LDX　送变址寄存器Y　　　LDX

操作：M→X

N	Z	C	I	D	V
✓	✓	—	—	—	—

寻址方式	汇编语言格式	操作码	字节数	周期数
立即型	LDX　#oper	A2	2	2
零　页	DDX　oper	A6	2	3
零页Y变址型	LDX　oper, Y	B6	2	4
绝对型	LDX　oper	AE	3	4
绝对Y变址型	LDX　oper, Y	BE	3	4*

* 当跨页时则加1。

LDY　送变址寄存器　　　LDY

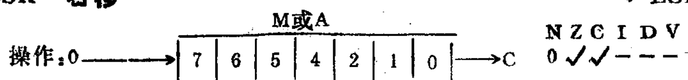
操作：M←Y

N	Z	C	I	D	V
✓	✓	—	—	—	—

寻址方式	汇编语言格式	操作码	字节数	周期数
立即型	LDY　#oper	A0	2	2
零　页	LDY　oper	A4	2	3
零页X变址型	LDY　oper, X	B4	2	4
绝对型	LDY　oper	AC	3	4
绝对X变址型	LDY　oper, X	BC	3	4*

* 当跨页时加1。

LSR 右移



寻址方式	汇编语言格式	操作码	字节数	周期数
累加器型	LSR A		1	2
零 页	LSR oper	46	2	5
零页X变址型	LSR oper, X	56	2	6
绝 对 型	LSR oper	4E	3	6
绝对X变址型	LSR oper, X	5E	3	7

NOP 空操作

N Z C I D V
— — — — — —

操作：空操作（2 周期）

寻址方式	汇编语言格式	操作码	字节数	周期数
隐 含 型	NOP	EA	1	2

ORA 逻辑或

N Z C I D V
✓ ✓ — — — —

操作：A∨M→A

寻址方式	汇编语言格式	操作码	字节数	周期数
立 即 型	ORA #oper	09	2	2
零 页	ORA oper	05	2	3
零页X变址型	ORA oper, X	15	2	4

(续)

寻址方式	汇编语言格式	操作码	字节数	周期数
绝对型	ORA oper	0D	3	4
绝对X变址型	ORA oper, X	1D	3	4*
绝对Y变址型	ORA oper, Y	19	3	4*
变址间接型	ORA (oper, X)	01	2	6
间接变址型	ORA (oper), Y	11	2	5*

*有跨页时则加1。

PHA 累加器压入堆栈 PHA

操作：A↓

N Z C I D V
— — — — — —

寻址方式	汇编语言格式	操作码	字节数	周期数
隐含型	PHA	48	1	3

PHP 处理器状态压入堆栈 PHP

操作：P↓

N Z C I D V
— — — — — —

寻址方式	汇编语言格式	操作码	字节数	周期数
隐含型	PHP	08	1	3

PLA 堆栈弹回累加器 PLA

操作：A↑

N Z C I D V
✓ ✓ – – – –

寻址方式	汇编语言格式	操作码	字节数	周期数
隐 含 型	PLA	68	1	4

PLP 堆栈弹回处理器状态 PLP

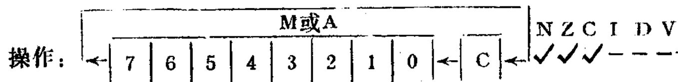
操作：P↑

N Z C I D V
来自堆栈

寻址方式	汇编语言格式	操作码	字节数	周期数
隐 含 型	PLP	28	1	4

ROL 循环左移 ROL

操作：

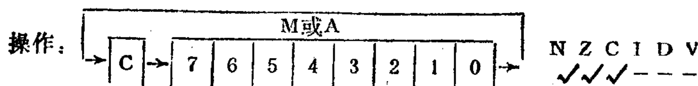


N Z C I D V
✓ ✓ ✓ – – –

寻址方式	汇编语言格式	操作码	字节数	周期数
累加器型	ROL A	2A	1	2
零 页	ROL oper	26	2	5
零页X变址型	ROL oper, X	36	2	6
绝 对 型	ROL oper	2E	3	6
绝对X变址型	ROL oper, X	3E	3	7

ROR 循环右移

操作：



N Z C I D V
✓✓✓ – – –

寻址方式	汇编语言格式	操作码	字节数	周期数
累加器型	ROR A	6A	1	2
零页	ROR oper	66	2	5
零页X变址型	ROR oper, X	76	2	6
绝对型	ROR oper	6E	3	6
绝对X变址型	ROR oper, X	7E	3	7

RTI 中断返回

N Z C I D V
来自堆栈

操作：P↑ PC↑

寻方址式	汇编语言格式	操作码	字节数	周期数
隐含型	RTI	40	1	6

RTS 子程序返回

N Z C I D V
– – – – – –

操作：PC↑，PC+1→PC

寻址方式	汇编语言格式	操作码	字节数	周期数
隐含型	RTS	60	1	6

SBC 带借位的减法

操作：A—M—$\overline{C}$→A

N	Z	C	I	D	V
✓	✓	✓	—	—	✓

注：$\overline{C}$借位

寻址方式	汇编语言格式	操作码	字节数	周期数
立即型	SBC *oper	E9	2	2
零页	SBC oper	E5	2	3
零页X变址型	SBC oper，X	F5	2	4
绝对型	SBC oper	ED	3	4
绝对X变址型	SBC oper，X	FD	3	4*
绝对Y变址型	SBC oper，Y	F9	3	4*
变址间接型	SBC (oper，X)	E1	2	6
间接变址型	SBC (oper)，Y	F1	2	5

* 当跨页时则加1。

SEC 置进位标志

操作：1→C

N	Z	C	I	D	V
—	—	1	—	—	—

寻址方式	汇编语言格式	操作码	字节数	周期数
隐含型	SEC	38	1	2

SED 置十进制方式

操作：1→D

N	Z	C	I	D	V
—	—	—	—	1	—

寻址方式	汇编语言格式	操作码	字节数	周期数
隐含型	SED	F8	1	2

SEI 置中断禁止位 SEI

操作：1→I

N Z C I D V
— — — 1 — —

寻址方式	汇编语言格式	操作码	字节数	周期数
隐含型	SEI	78	1	2

STA 存累加器 STA

操作：A→M

N Z C I D V
— — — — — —

寻址方式	汇编语言格式	操作码	字节数	周期数
零页	STA oper	85	2	3
零页X变址型	STA oper, X	95	2	4
绝对	STA oper	8D	3	4
绝对X变址型	STA oper, X	9D	3	5
绝对Y变址型	STA oper, Y	99	3	5
变址间接型	STA (oper, X)	81	2	6
间接变址型	STA (oper), Y	91	2	6

STX 存变址寄存器X STX

操作：X→M

N Z C I D V
— — — — — —

寻址方式	汇编语言格式	操作码	字节数	周期数
零　　页	STX　oper	86	2	3
零页Y变址型	STX　oper, Y	96	2	4
绝　对　型	STX　oper	8E	3	4

STY　存变址寄存器 Y　　STY

操作：Y→M

N Z C I D V
— — — — — —

寻址方式	汇编语言格式	操作码	字节数	周期数
零　　页	STY　oper	84	2	3
零页X变址型	STY　oper, X	94	2	4
绝　对　型	STY　oper	8C	3	4

TAX　累加器送变址寄存器 X　　TAX

操作：A→X

N Z C I D V
√ √ — — — —

寻址方式	汇编语言格式	操作码	字节数	期周数
隐　含　型	TAX	AA	1	2

TAY 累加器送变址寄存器 Y TAY

操作：A→Y

N Z C I D V
✓ ✓ — — — —

寻址方式	汇编语言格式	操作码	字节数	周期数
隐 含 型	TAY	A8	1	2

TYA 变址寄存器 Y 送累加器 TYA

操作：Y→A

N Z C I D V
✓ ✓ — — — —

寻址方式	汇编语言格式	操作码	字节数	周期数
隐 含 型	TYA	98	1	2

TSX 栈指针送变址寄存器 X TSX

操作：S→X

N Z C I D V
✓ ✓ — — — —

寻址方式	汇编语言格式	操作码	字节数	周期数
隐 含 型	TSX	BA	1	2

TXA 变址寄存器X送累加器 TXA

操作：X→A

N	Z	C	I	D	V
✓	✓	—	—	—	—

寻址方式	汇编语言格式	操作码	字节数	周期数
隐　含　型	TXA	8A	1	2

TXS 变址寄存器X送栈指针 TXS

操作：X→S

N	Z	C	I	D	V
—	—	—	—	—	—

寻址方式	汇编语言格式	操作码	字节数	周期数
隐　含　型	TXS	9A	1	2

附录 B 字母、字符显示码表(基本字符组)

	反向显示		闪烁显示		正向显示		正向显示	
	大写字母	特殊字符	大写字母	特殊字符	大写字母	特殊字符	大写字母	小号字母
	$ 00-$ 1F	$ 20-$ 3F	$ 40-$ 5F	$ 60-$ 7F	$ 80-$ 9F	$ A0-$ BF	$ C0-$ DF	$ E0-$ FF
00	@		@		@		@	`
01	A	!	A	!	A	!	A	a
02	B	"	B	"	B	"	B	b
03	C	#	C	#	C	#	C	c
04	D	$	D	$	D	$	D	d
05	E	%	E	%	E	%	E	e
06	F	&	F	&	F	&	F	f
07	G	'	G	'	G	'	G	g
08	H	(	H	(	H	(	H	h
09	I	)	I	)	I	)	I	i
0A	J	*	J	*	J	*	J	j
0B	K	+	K	+	K		K	k
0C	L	,	L	,	L	,	L	l
0D	M	—	M	—	M	—	M	m
0E	N	.	N	.	N	.	N	n
0F	O	/	O	/	O	/	O	o
10	P	ϕ	P	ϕ	P	ϕ	P	p
11	Q	1	Q	1	Q	1	Q	q
12	R	2	R	2	R	2	R	r

续表

	反向显示		闪烁显示		正向显示		正向显示	
	大写字母	特殊字符	大写字母	特殊字符	大写字母	特殊字符	大写字母	小写字母
	$00-$1F	$20-$3F	$40-$5F	$60-$7F	$80-$9F	$A0-$BF	$C0-$DF	$E0-$FF
13	S	3	S	3	S	3	S	s
14	T	4	T	4	T	4	T	t
15	U	5	U	5	U	5	U	u
16	V	6	V	6	V	6	V	v
17	W	7	W	7	W	7	W	w
18	X	8	X	8	X	8	X	x
19	Y	9	Y	9	Y	9	Y	y
1A	Z	:	Z	:	Z	:	Z	z
1B	[	;	[	;	[	;	[	{
1C	\	<	\	<	\	<	\	|
1D	]	=	]	=	]	=	]	}
1E	∧	>	∧	>	∧	>	∧	~
1F	—	?	—	?	—	?	—	■